Parameter selection in lattice-based cryptography

Rachel Player

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
Royal Holloway, University of London

2017

Declaration

These doctoral studies were conducted under the supervision of Prof. Carlos Cid and Prof. Sean Murphy.

The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Department of Information Security as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Rachel Player
2 November 2017

Abstract

Public-key cryptography in use today is based on classically hard problems such as factoring or solving discrete logarithms. These problems could be efficiently solved if an adversary had access to a sufficiently large quantum computer. The potential of such a quantum computer therefore represents a threat on current cryptography. The field of post-quantum cryptography aims to mitigate against this threat by proposing schemes based on alternative assumptions that are believed to be hard in both the classical and quantum setting.

Lattice-based cryptography has emerged as a promising candidate for post-quantum cryptography. One reason for this is the wealth of applications that are possible, perhaps the most notable of which is Fully Homomorphic Encryption (FHE). This enables computations to be performed on encrypted data, without requiring access to the secret key, and for these computations to correspond to operations on the underlying data in a meaningful way.

The Learning with Errors (LWE) problem and its variants, such as LWE with small secret, LWE with binary error, and Ring-LWE, are used as hardness assumptions in many lattice-based schemes. In this thesis we consider parameter selection in cryptosystems based on LWE. We begin with a focus on security by considering the concrete hardness of LWE. We comprehensively review the algorithms that can be used to solve LWE and its variants with a small secret. Turning our attention to an LWE variant where the error distribution is binary, we show there is an additional attack applicable in this setting.

In applications, the selection of appropriate parameters is often very challenging due to the conflicting requirements of security, correctness and performance. We highlight this in the application setting of FHE by considering a scheme based on Ring-LWE. In particular, we discuss the selection of parameters in SEAL, an implementation of the scheme by Fan and Vercauteren.

Contents

List of Tables

List of Figures

Publications

Parts of this thesis are based on the following works.

- Martin R. Albrecht, Rachel Player and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.
- Johannes Buchmann, Florian Göpfert, Rachel Player and Thomas Wunderer. On the Hardness of LWE with Binary Error: Revisiting the Hybrid Lattice-Reduction and Meet-in-the-Middle Attack. In D. Pointcheval, A. Nitaj, and T. Rachidi, editors, *AFRICACRYPT 2016*, volume 9646 of Lecture Notes in Computer Science, pages 24–43. Springer, 2016.
- Hao Chen, Kim Laine and Rachel Player. Simple Encrypted Arithmetic Library - SEAL. In M. Brenner and K. Rohloff, editors, *WAHC'17 - 5th Workshop on Encrypted Computing and Applied Homomorphic Cryptography*, to appear.
- Sean Murphy and Rachel Player. Noise Distributions in Homomorphic Ring-LWE. *In submission.*
- Kim Laine and Rachel Player. Simple Encrypted Arithmetic Library - SEAL (v2.0). *Technical report, September 2016.*
- Hao Chen, Kim Laine and Rachel Player. Simple Encrypted Arithmetic Library - SEAL (v2.1). *Technical report, September 2016.*
- Hao Chen, Kim Laine and Rachel Player. Simple Encrypted Arithmetic Library - SEAL (v2.2). *Technical report, June 2017.*

The following work was also undertaken during the these doctoral studies.

- Hao Chen, Kim Laine, Rachel Player and Yuhou Xia. High-Precision Arithmetic in Homomorphic Encryption. *In submission.*

Acknowledgements

As I was warned, writing this page has been the hardest part.

All of this would have been impossible without the guidance of my supervisors Carlos Cid and Sean Murphy, and the financial aid of an ACE-CSR PhD grant, which I gratefully acknowledge. Carlos, thank you for allowing me the freedom to follow my interests, and of course, the trip to Japan! Sean, thank you for your patience with both my naive questions about statistics and the many emails of my unfiltered thinking out loud.

Many thanks must also go to Martin Albrecht, who has enabled me to access many opportunities that have enriched me as a researcher, and from whom I have learnt a great deal. Thank you also to Kenny Paterson, for the career advice, and retweets!

Thank you to Prof. Buchmann, Florian Göpfert and Thomas Wunderer for hosting me during a productive research visit in Darmstadt. Florian deserves an additional thank you for pointing out several typos in [16], which are fixed here.

A huge thank you to Kristin Lauter and Kim Laine for having me as an intern at Microsoft Research and again for a research visit. Working with the MSR team has been an incredibly motivating experience and I am very grateful for the opportunity.

Team Llama: Christian, Dan, James and Naomi, vielen Dank for the many fun Tuesdays and other adventures. Thanks to Alex, Amit, Ben, Dale, Eamonn, Ela, Fernando, Gordon, Greg, Konstantinos, Joanne, Lydia, Marie-Sarah, Rob, Sheila, Susan, Favourite Thalia, Thyla and everyone else past and present at RHUL for being so friendly. A special thank you to Sam for being an awesome co-author. To everyone in the WISDOM group: I am so proud of us!

Thank you to the whole Bristol group, especially Ana, for being my conference bestie.

I am so fortunate to have so many good friends. Savvy, thank you for teaching me pro level networking and for always knowing what to say. Ben, thank you for countless injokes which have been essential during these hashtag four more years! Beautiful Soph, thank you for tolerating all the anxious texts and for always laughing at my jokes. Catherine, Darren, Miranda, Pete, Warwick Wheatley and the rest of the Ktown family, thank you all for the cups of tea and the many trips not to Nottingham.

To my whole family, I am so grateful for your endless love and support throughout all my adventures. I dedicate this thesis in loving memory of my grandpa, Donald Freeman, who encouraged me in maths from the very beginning.

Finally, James, thank you for always being by my side - it means more than I can say. I have a longer, more gushing paragraph, which this page is too small to contain!

Chapter 1

Introduction

Contents

This chapter gives an overview of the thesis and its contributions. We provide the motivation for our research and describe the structure of the thesis.

1.1 Motivation

Current public-key cryptography is based on problems such as factoring or solving discrete logarithms. These problems have been widely studied and are believed to be hard to solve on a classical computer. However, an adversary equipped with a sufficiently large quantum computer can solve these problems easily using Shor's algorithm [203]. Such a quantum computer does not exist today; nonetheless, its potential is considered such a threat that, since 2016, NIST is seeking to standardise quantum-resistant public-key cryptographic algorithms [180].

The field of post-quantum cryptography, encompassing schemes based on various alternative assumptions that are believed to be hard in both the classical and quantum setting, has received increasing attention in recent years (although some schemes, such as [162, 164], now falling under this umbrella have been considered for longer, and even predate Shor's algorithm). There are five main sub-fields of post-quantum cryptography: multivariate cryptography, hash-based cryptography, cryptography based on supersingular elliptic curve isogenies, code-based cryptography, and lattice-based cryptography. The work in this thesis falls into this final area, mainly focussing on assumed hard problems underlying lattice-based cryptography, but also consider-

ing an application made possible in the lattice setting.

Lattice-based cryptography has become popular in recent years for several reasons in addition to its potential use in a post-quantum world. A key reason dates back to the work of Ajtai [5] who linked the average case complexity of lattice problems to the worst case, showing that solving certain lattice problems in the average case is at least as hard as solving approximate shortest vector problem in the worst case. Another reason is the wealth of applications of lattice-based cryptography (see the survey [187]). These include (Hierarchical) Identity Based Encryption [54, 111], Attribute-Based Encryption [49], Oblivious Transfer schemes [190], Circular-Secure Encryption [23], and Leakage-Resilient Encryption [114].

Fully Homomorphic Encryption (FHE) is perhaps the flagship application of lattice-based cryptography. FHE enables computations to be performed on encrypted data, without requiring access to the secret key, and for these computations to correspond to operations on the underlying data in a meaningful way. First proposed by Rivest *et al.* [198] in 1978, it remained a longstanding open problem to construct such a scheme. This was finally resolved by Gentry [106] in 2009, who proposed a Fully Homomorphic Encryption scheme based on problems in ideal lattices.

The Learning with Errors problem (LWE), introduced in 2005 by Regev [194], is a presumed hard problem underlying many lattice-based schemes. A prominent variant of LWE, which we discuss in detail in Chapter 6, is the Ring Learning with Errors problem (Ring-LWE) [210, 157]. Ring-LWE is widely preferred for its smaller key sizes compared with LWE. Other presumed hard problems used in lattice-based cryptography include the Short Integer Solutions problem (SIS) [5], which can be seen as dual to LWE [165]; and the Approximate Greatest Common Divisor problem (AGCD) [132], which has a reduction from LWE [71]. A prominent lattice-based scheme is the NTRU encryption scheme [131], a variant of which is based on Ring-LWE [209]. It is therefore clear that LWE is a central problem in lattice-based cryptography.

Besides Ring-LWE, many other variants of LWE have been proposed in the literature [183, 47, 191, 102, 166, 45, 147, 67, 199]. Work in this direction is often motivated by the needs of applications: for example, the use of a small secret for

efficiency reasons [119], or a modified error distribution, for ease of sampling [19]. Confidence in some variants is bolstered by reductions from standard lattice problems [157, 166, 189]; indeed, certain variants have been introduced as intermediates in reductions [45, 67].

For several variants of LWE it is the case that more attacks apply [29, 10, 8], sometimes so devastating that the variant would be considered easy in settings where the analogous standard LWE instance would be considered infeasible to solve [129]. Motivated by identifying what would characterise an easy LWE variant, the line of work [97, 99, 62, 145, 63, 56, 55, 188] has considered the limitations of the use of LWE as a hardness assumption. In Chapter 5 we augment the literature, focussing on LWE with uniform binary error [166, 10]. We show that this variant is easier than expected, by exhibiting an additional attack that applies in this setting.

A common criticism of lattice-based cryptography is that it is challenging to select secure parameters for lattice-based schemes. This thesis aims to address this issue, focussing on schemes based on LWE. Such schemes are very often supported by a security reduction from the underlying LWE problem. An adversary breaking a scheme based on LWE can be used to solve the underlying LWE problem, possibly with some extra effort, which corresponds to the so-called tightness gap [57, 17]. The tightness gap, in turn, corresponds to a difference in the bit security of the scheme versus the bit hardness of the underlying LWE problem. In practice this issue is overwhelmingly ignored and designers of schemes typically select parameters to ensure a certain level of security (such as 80 or 128 bits) based solely on the underlying LWE instance: that is, the LWE instance is expected to take 2^{80} or 2^{128} operations respectively to solve. In Chapters 4 and 5 we therefore focus on the concrete hardness of the Learning with Errors problem. We comprehensively review the algorithms that can be used to solve LWE and its variants with small secret.

When selecting parameters in applications, we must consider not only security, but also correctness and performance. In many application settings these requirements can be conflicting. In Chapter 7, we highlight these issues in the setting of Fully Homomorphic Encryption. For example, on the one hand, the larger the value of the modulus q, the larger the number of homomorphic evaluation operations that can be performed, while on the other hand, the larger the value of q, the lower the

security. We discuss the selection of parameters in the homomorphic encryption library SEAL [93], which implements the scheme of Fan and Vercauteren [100].

1.2 Structure of thesis

The remainder of the thesis is organised as follows.

Chapter 2. This chapter defines notation that will be used throughout the thesis, and introduces lattices and the Learning with Errors problem (LWE).

Chapter 3. This chapter gives a comprehensive review of lattice basis reduction algorithms, which are an essential step in many approaches for solving LWE. Almost all of the material is from the literature, and the chapter is meant as an extended background on the topic. However, we also present a new result on the asymptotic time complexity of the BKZ algorithm. This chapter is derived from material presented in [16].

Chapter 4. This chapter reviews strategies and algorithms in the literature for solving LWE. The material presented is an extended and updated version of material presented in [16]. We provide evidence that the widely-used Lindner and Peikert [150] model for the cost of lattice reduction is inaccurate. In this chapter and the previous chapter, we give an up-to-date account of the LWE estimator of Albrecht [6], which was developed alongside the paper [16].

Chapter 5. This chapter considers variants of LWE with unusually small secret or error. We begin with a discussion of small secret LWE, extending and updating material presented in [16].

The remainder of the chapter is based on material presented in [52]. We focus on the variant of LWE with binary error and show that the hybrid attack, introduced by Howgrave-Graham [133] as an attack on NTRU, can be applied in this setting. We

adapt algorithms discussed earlier in the chapter, and in the previous chapter, to the binary error setting. A comparison of the hybrid attack with other approaches was presented in [52], but used the Lindner and Peikert [150] model for the cost of lattice reduction. We provide an updated comparison using a more realistic cost model.

Chapter 6. This chapter considers the Ring Learning with Errors problem (Ring-LWE) and provides relevant algebraic and statistical background material that will be required in the following chapter, based on material presented in [173]. We also review a result of Ducas and Durmus [95], highlighting and correcting an error in their proof: this is based on joint (unpublished) work with Cid and Murphy.

Chapter 7. This chapter considers homomorphic encryption schemes based on variants of Ring-LWE. We begin by highlighting two general issues in such schemes, namely security and encoding. We then turn our attention to parameter selection in SEAL [93], a homomorphic encryption library developed by Microsoft Research that the author became involved with during an internship [146, 58, 59, 60]. The main task undertaken by the author during the internship was to implement the FV scheme [100] which was released as SEAL v2.0 [146]. Previously, SEAL had been based on the YASHE scheme [41]. We give a thorough analysis of noise growth behaviour, considering both inherent and invariant noise, and argue that invariant noise is more natural. We discuss parameters affecting security, and parameters affecting the relinearization operation, which impact on noise growth and the running time of key generation. The chapter is based on material presented in [146, 58, 59, 60].

1.3 Original work and our individual contributions

The material in Chapter 3, Chapter 4 and the first half of Chapter 5 is based on the paper [16], which was a joint work of Albrecht, myself and Scott. I wrote the first draft of most sections of [16], except [16, Section 5.4], a version of which appears here as Section 4.7; and [16, Section 7 and Section 8], which are omitted here. Albrecht then edited the draft sections I wrote, in some cases majorly. All authors contributed to later stages of editing of the document. The paper [16] was a survey and so is

largely not original, although the work does include the following contributions: an asymptotic bound on the cost of BKZ, the details of a Meet-in-the-Middle algorithm for solving LWE, evidence that the Lindner and Peikert [150] cost model for BKZ is inaccurate, and the LWE estimator [6] sage tool. The last two of these contributions are due to Albrecht.

The analysis of a Meet-in-the-Middle algorithm, presented here in Section 4.3.1, was developed for [16] through joint discussions among all the authors, and has been refined for the thesis by Albrecht and myself. The asymptotic bound on BKZ is due to myself. The version presented here in Section 3.2.1 is a more clear formulation of the original [16, Lemma 5]. Other parts of Chapter 3, Chapter 4 and the first half of Chapter 5 extend or update [16]. The case $\lambda_2(L(\mathbf{B})) = q$, discussed in Section 4.8, provides the analysis required to fix an open issue[1] in [6] and is an additional contribution of the thesis.

The material in the second half of Chapter 5 is based on the paper [52], which was a joint work of Buchmann, Göpfert, myself and Wunderer. The main idea of the paper [52], that the hybrid attack would apply to LWE instances with binary error, was developed jointly during a research visit in Prof. Buchmann's group. A theoretical analysis of the hybrid attack given in [52, Section 3.2] is due to Göpfert and Wunderer, so the details are omitted here. Adapting other algorithms for the case of LWE with binary error, presented here in Sections 5.7, 5.8 and 5.9, is another original contribution of [52], due to Göpfert and myself. For other parts of the paper [52], Göpfert, myself and Wunderer contributed equally, while Prof. Buchmann acted in a supervisory role and suggested improvements to the structure and motivation of early drafts of the paper.

The comparison to other approaches for solving LWE with binary error, presented here in Section 5.11, improves on the comparison presented in [52, Section 4] as it no longer uses the Lindner and Peikert cost model for BKZ, and is an additional contribution of the thesis. Correspondingly to the general LWE case, the Meet-in-the-Middle algorithm presented in Section 5.7 has been updated compared to [52] and is an additional contribution of the thesis.

[1] `https://bitbucket.org/malb/lwe-estimator/issues/4/lambda_2-not-implemented`

1.3 Original work and our individual contributions

Most of the material in Chapter 6 is derived from parts of the paper [173], which is a joint work with Murphy. In particular, the chapter is an extended version of [173, Sections 1 and 2], which were largely written by myself. The material presented in Section 6.3 is based on [173, Section 5], and is due to Murphy. The original material in [173] consists of statistical results due to Murphy that are omitted here. Section 6.6 arose from discussions with Cid and Murphy and contains material based on original drafts by Murphy and myself. This is the only part of the chapter containing original work: namely, the correction to the proof [95, Theorem 5].

The material in Chapter 7 is based on the paper [146], which was a joint work of Laine and myself, and the papers [58, 59, 60], which were joint works of Chen, Laine and myself. The technical report [146] describes the implementation of the FV scheme in SEAL [93] to form SEAL v2.0, which is work I undertook during an internship at Microsoft Research. I drafted the paper [146] and it was later edited and improved by Laine and myself.

Improvements to SEAL were later released as SEAL v2.1 [58] and v2.2 [60]. The technical reports [58, 60] are extensions of [146] reporting on these improvements and in general the sections extending [146] were not written by me, since I have not been involved in the development of code for SEAL since v2.0. However, during a subsequent research visit to Microsoft Research, I was involved with producing the invariant noise estimates described in [60], which are presented here in Section 7.4.2 and arose from joint discussions with Chen and Laine. The definition of invariant noise is a new notion of noise for FV and an original contribution of [60].

The publication [59] is derived from [58] with a new section on related work and and extended discussion on security, which were drafted by myself, and subsequently edited by all authors. These sections, which have themselves been extended for the thesis, form the basis of Sections 7.1.1 and 7.2 respectively.

The full proofs for the noise bounds, presented in Section 7.4 have not previously been published and are additional contributions of the thesis. The discussion on relinearization, presented in Section 7.6, has also not previously been published. This section arose from joint discussions with Laine.

Chapter 2

Notation and background

Contents

This chapter establishes notation used throughout the thesis, and provides background material.

2.1 Notation

We denote vectors in bold: for example, $\mathbf{a}$; and matrices in upper-case bold: for example, $\mathbf{A}$. By $\mathbf{a}_{(i)}$ we denote the i^{th} component of $\mathbf{a}$; that is, a scalar. In contrast, $\mathbf{a}_i$ denotes the i^{th} element of a list of vectors. For a vector $\mathbf{a} \in \mathbb{R}^n$ we write $\mathbf{a} \mod q$ for its unique representative modulo q in $[-\lfloor \frac{q}{2} \rfloor, \frac{q}{2})^n$. We denote by $\langle \cdot, \cdot \rangle$ the usual dot product of two vectors and by $\langle \cdot, \cdot \rangle_p$ this dot product modulo p.

The base-a logarithm of x is denoted $\log_a(x)$ except for the natural logarithm of x which is denoted $\ln(x)$. Logarithms are base 2 if no base is explicitly stated.

The complex conjugate of z is denoted $\overline{z}$. We denote by $\mathbf{A}^\dagger$ the complex conjugate transpose of the matrix $\mathbf{A}$, so $\mathbf{A}^\dagger = \overline{\mathbf{A}}^T$.

The expectation of a random variable X is denoted $\mathrm{E}[X]$ and the variance is denoted $\mathrm{Var}[X]$. A Gaussian or Normal distribution with mean μ and standard deviation σ

is denoted $N(\mu, \sigma)$. A complex Normal distribution with mean μ, covariance matrix $\mathbf{\Gamma}$ and relation matrix $\mathbf{C}$ is denoted $\mathbb{C}\mathrm{N}(\mu, \mathbf{\Gamma}, \mathbf{C})$ and is real if and only if $\mathbf{\Gamma} = \mathbf{C}$.

By $a \xleftarrow{\$} \mathcal{S}$ we denote that a is sampled uniformly at random from the finite set $\mathcal{S}$. By $e \leftarrow \chi$ we denote that e is sampled according to the distribution χ. Such sampling is independent of any other sampling.

For a ring R and an integer q, we denote by R_q the quotient ring R/qR. Euler's totient function is denoted $\varphi(m)$. The tensor product of the vector spaces V and W over a field K is denoted $V \otimes_K W$.

We use $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ to denote respectively rounding down and up to the nearest integer. We use $\lfloor \cdot \rceil$ to denote rounding to the nearest integer, rounding up in case of ambiguity. When these operations are applied to a polynomial, we mean performing the corresponding operation to each coefficient separately. The operation $[\cdot]_t$ denotes the reduction of an integer or a polynomial modulo an integer t. When applied to polynomials, the reduction modulo t is applied to every integer coefficient separately. The reductions are always done into the symmetric interval $\left[-\frac{t}{2}, \frac{t}{2}\right)$.

The norm $\| \cdot \|$ always denotes the infinity norm, while the usual Euclidean norm is denoted $\| \cdot \|_2$. The norm $\| \cdot \|_1$ denotes the ℓ_1 norm. The norm $\| \cdot \|^{\mathrm{can}}$ denotes the *canonical embedding norm*, which will be defined in Chapter 7.

A negligible function $\mathrm{negl} : \mathbb{N} \to \mathbb{R}$ is a function such that for all $c \in \mathbb{N}$, there exists $N_c \in \mathbb{N}$ such that for all $n > N_c$, $|\mathrm{negl}(n)| < \frac{1}{n^c}$. The error function (see [3]) of a value $x \geq 0$ is denoted $\mathrm{erf}(x)$. We write $2 \leq \omega < 3$ for any constant such that, for sufficiently large n, there is an algorithm that multiplies two $n \times n$ matrices in $\mathcal{O}(n^\omega)$ arithmetic operations.

2.2 Lattice background

A *lattice* L is a discrete additive subgroup of $\mathbb{R}^m$. We always consider a lattice in terms of a specified basis $\mathbf{B} = \{\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}\}$ of n linearly independent vectors where $n \leq m$. The lattice $L(\mathbf{B})$ is the set of linear combinations of the basis vectors with

integer coefficients:

$$L(\mathbf{B}) = \left\{ \sum_{i=0}^{n-1} x_i \mathbf{b}_i \;\middle|\; x_i \in \mathbb{Z} \right\} .$$

We slightly abuse notation and let $\mathbf{B}$ also denote the matrix whose rows are the basis vectors. The *rank* of the lattice L is defined to be the rank of the matrix $\mathbf{B}$. If the rank equals m we say that L is *full-rank*. For a full-rank lattice L, the *dual lattice* is defined as

$$L^* = \{ \mathbf{v} \in \mathbb{R}^m \,|\, \forall \mathbf{w} \in L,\, \langle \mathbf{w}, \mathbf{v} \rangle \in \mathbb{Z} \} .$$

We will very often be concerned with lattices L such that $q\mathbb{Z}^m \subseteq L \subseteq \mathbb{Z}^m$: these are known as *$q$-ary lattices*. Every q-ary lattice is full-rank.

A *unimodular* matrix is a square integer matrix having determinant ± 1. There are infinitely many lattice bases for $n > 1$, and it is a standard result that two bases generate the same lattice if they are related by a unimodular matrix in the following way.

Lemma 1. *Let* $\mathbf{B}$ *and* $\mathbf{B}'$ *be two bases. Then* $L(\mathbf{B}') = L(\mathbf{B})$ *if and only if* $\mathbf{B}' = \mathbf{U}\mathbf{B}$ *where* $\mathbf{U}$ *is a unimodular matrix.*

The *determinant* or *volume* $\mathrm{vol}\,(L)$ of a full-rank lattice L is the absolute value of the determinant of any basis of the lattice. This can be seen to be an invariant of the lattice: suppose $\mathbf{B}$ and $\mathbf{B}'$ are two bases of the lattice, then, by Lemma 1, $\mathbf{B}' = \mathbf{U}\mathbf{B}$ for some unimodular matrix $\mathbf{U}$. So, since unimodular matrices have determinant ± 1,

$$|\det(\mathbf{B}')| = |\det(\mathbf{U}\mathbf{B})| = |\det(\mathbf{U}) \cdot \det(\mathbf{B})| = |\det(\mathbf{U})| \cdot |\det(\mathbf{B})| = |\det(\mathbf{B})| .$$

The *fundamental parallelepiped* of a lattice $L(\mathbf{B})$ where $\mathbf{B} = \{\mathbf{b}_0, ..., \mathbf{b}_{n-1}\}$ is given by

$$\mathcal{P}(\mathbf{B}) = \left\{ \mathbf{x} \in \mathbb{R}^m \mid \mathbf{x} = \sum_{i=0}^{n-1} \alpha_i \mathbf{b}_i \text{ for } -\frac{1}{2} \leq \alpha_i < \frac{1}{2} \right\} .$$

An equivalent definition of the volume $\mathrm{vol}\,(L)$ of a lattice $L = L(\mathbf{B})$ is the volume of its fundamental parallelepiped $\mathcal{P}(\mathbf{B})$.

The i^{th} *successive minimum* of a lattice, $\lambda_i(L)$, is the radius of the smallest ball centred at the origin containing at least i linearly independent lattice vectors. In particular, $\lambda_1(L)$ is the length of a shortest nonzero lattice vector.

Definition 1. *The* Shortest Vector Problem *(SVP) is defined as follows. Given a basis* $\mathbf{B}$ *of a lattice* $L = L(\mathbf{B})$*, find a vector* $\mathbf{v} \in L$ *such that* $\|\mathbf{v}\|_2 = \lambda_1(L)$*.*

We now define a series of problems related to SVP. Each is parameterised by an approximation factor $\gamma > 1$.

Definition 2. *The* γ-Approximate Shortest Vector Problem *(*SVP_γ*) is defined as follows. Given a basis* $\mathbf{B}$ *of a lattice* $L = L(\mathbf{B})$*, find a nonzero vector* $\mathbf{v} \in L$ *such that* $\|\mathbf{v}\|_2 \leq \gamma \cdot \lambda_1(L)$*.*

Definition 3. *The* γ-Gap Shortest Vector Problem *(*$GapSVP_\gamma$*) is defined as follows. Given a basis* $\mathbf{B}$ *of a lattice* $L = L(\mathbf{B})$ *and a real number* $d > 0$*, return YES if* $\lambda_1(L) \leq d$ *and NO if* $\lambda_1(L) > \gamma \cdot d$*. If* $d < \lambda_1(L) \leq \gamma \cdot d$*, there are no requirements on the output.*

Definition 4. *The* γ-Shortest Independent Vector Problem *(*$SIVP_\gamma$*) is defined as follows. Given a basis* $\mathbf{B}$ *of a lattice* $L = L(\mathbf{B})$*, find* n *linearly independent vectors* $\mathbf{v}_1, \ldots, \mathbf{v}_n$ *such that* $\max(\|\mathbf{v}_i\|_2) \leq \gamma \cdot \lambda_n(L)$*.*

Definition 5. *The* γ-unique Shortest Vector Problem *(*$uSVP_\gamma$*) is defined as follows. Given a basis* $\mathbf{B}$ *of a lattice* $L = L(\mathbf{B})$ *such that* $\lambda_2(L) > \gamma\lambda_1(L)$*, find a vector* $\mathbf{v} \in L$ *such that* $\|\mathbf{v}\|_2 = \lambda_1(L)$*.*

It may be difficult to verify a solution to approximate SVP in a lattice L as even the length $\lambda_1(L)$ itself may not be known [104]. In contrast, the volume $\mathrm{vol}(L)$ can be computed easily from any basis. By Minkowski's second theorem [172],

$$\lambda_1(L) \leq \sqrt{\gamma_n} \cdot \mathrm{vol}(L)^{\frac{1}{n}} ,$$

where γ_n is Hermite's constant [126] in dimension n. This motivates the following problem.

Definition 6. *The* γ-Hermite Shortest Vector Problem *(*γ*-HSVP) is defined as follows. Given a basis* $\mathbf{B}$ *of a lattice* $L = L(\mathbf{B})$*, find a nonzero vector* $\mathbf{v} \in L$ *such that* $\|\mathbf{v}\|_2 \leq \gamma \cdot \mathrm{vol}(L)^{\frac{1}{n}}$*.*

The *Gaussian heuristic* states that for a measurable subset $K \subset \mathbb{R}^m$ and a lattice $L \subset \mathbb{R}^m$, we expect that the number of lattice points in K is approximately equal

to $\frac{\text{vol}(K)}{\text{vol}(L)}$. Specialising this to K being the unit Euclidean ball we can approximate $\lambda_1(L)$ as follows.

Heuristic 1. *The* Gaussian heuristic *states that*

$$\lambda_1(L) \approx \sqrt{\frac{m}{2\pi e}} \text{vol}\,(L)^{1/m} \ .$$

There are several occasions when we wish to find a lattice point close in Euclidean distance to a given point in space.

Definition 7. *Let L be a lattice and let $\mathbf{x}$ be a point. Let $\mathbf{y} \in L$ be the lattice point for which the length $\|\mathbf{x}-\mathbf{y}\|_2$ is minimised. We define the* distance to the lattice L *from the point $\mathbf{x}$ to be this length, which we denote dist$(\mathbf{x}, L)$.*

The *Closest Vector Problem* is to find a closest lattice vector to a target point in space. This can be seen as an inhomogeneous version of SVP, which could be formulated as finding a closest nonzero lattice vector to the point 0.

Definition 8. *The* Closest Vector Problem *(CVP) is defined as follows. Given a basis $\mathbf{B}$ of a lattice $L = L(\mathbf{B})$ and a target vector $\mathbf{t}$, find a vector $\mathbf{y} \in L$ such that $\|\mathbf{t}-\mathbf{y}\|_2$ is minimised.*

Again, for an approximation factor $\gamma > 1$, we can define two problems related to CVP.

Definition 9. *The* γ-Closest Vector Problem *is defined as follows. Given a basis $\mathbf{B}$ of a lattice $L = L(\mathbf{B})$ and a target vector $\mathbf{t}$, find a vector $\mathbf{y} \in L$ such that $\|\mathbf{t}-\mathbf{y}\|_2 < \gamma \cdot dist(\mathbf{t}, L)$.*

Definition 10. *The* γ-Bounded Distance Decoding *problem (BDD_γ) is defined as follows. Given a basis $\mathbf{B}$ of a lattice $L = L(\mathbf{B})$ and a target vector $\mathbf{t}$ such that $dist(\mathbf{t}, L) \leq \gamma \cdot \lambda_1(L)$, find the vector $\mathbf{y} \in L$ that is closest to $\mathbf{t}$.*

Babai's Nearest Plane algorithm [27] solves the approximate Closest Vector Problem. Its input is a lattice basis $\mathbf{B} \subset \mathbb{Z}^m$ and a target vector $\mathbf{t} \in \mathbb{R}^m$. Its output is a vector $\mathbf{e} \in \mathbb{R}^m$ such that $\mathbf{t}-\mathbf{e} \in L(\mathbf{B})$, which we denote by $\text{NP}_\mathbf{B}(\mathbf{t}) = \mathbf{e}$. If the basis $\mathbf{B}$ is clear from the context, we omit it in the notation and simply write $\text{NP}(\mathbf{t})$. Babai's

Nearest Plane algorithm works by recursively computing the closest vector on the sublattice spanned by subsets of the *Gram-Schmidt vectors* $\mathbf{b}_i^*$.

Definition 11. *For a lattice basis* $\mathbf{b}_0, \ldots, \mathbf{b}_{n-1} \in \mathbb{Z}^m$*, the* Gram-Schmidt vectors *are defined as* $\mathbf{b}_0^*, \ldots, \mathbf{b}_{n-1}^*$ *with*

$$\mathbf{b}_0^* = \mathbf{b}_0$$
$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=0}^{i-1} \mu_{i,j} \mathbf{b}_j^*$$

where $\mu_{i,j} = \frac{\langle \mathbf{b}_j^*, \mathbf{b}_i \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$ *are the* Gram-Schmidt coefficients.

As with a basis, the Gram-Schmidt vectors can be arranged in a matrix, which we denote $\mathbf{B}^*$. The product of the norm of the Gram-Schmidt vectors is equal to the volume of the lattice.

In general, the Gram-Schmidt vectors $\mathbf{b}_i^*$ do not themselves form a basis of the lattice $L(\mathbf{B})$. By an abuse of notation, for a matrix of Gram-Schmidt vectors $\mathbf{B}^*$ we define the parallelepiped $\mathcal{P}(\mathbf{B}^*)$ as for the definition of the fundamental parallelepiped. We will use the following result about the output of Babai's Nearest Plane algorithm.

Lemma 2 ([28])**.** *For a lattice basis* $\mathbf{B}$ *with Gram-Schmidt vectors* $\mathbf{B}^*$ *and a target vector* $\mathbf{t}$ *as input, Babai's Nearest Plane algorithm returns the unique vector* $\mathbf{e} \in \mathcal{P}(\mathbf{B}^*)$ *that satisfies* $\mathbf{t} - \mathbf{e} \in L(\mathbf{B})$*.*

As noted above, lattice bases are not unique, and certain bases may be preferable to others. In Chapter 3 we will discuss lattice basis reduction algorithms, which take as input a basis and output a *reduced basis* that can be considered more preferable. The quality of a reduced basis is characterised by the *Hermite factor* δ_0^n, which can be easily computed. We adopt the convention that the first nonzero vector, say $\mathbf{b}_0$, in a reduced lattice basis is a shortest vector in the basis.

Definition 12. *Let* $\{\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}\}$ *be a basis output by a lattice reduction algorithm. The* Hermite factor δ_0^n *is such that the shortest nonzero vector* $\mathbf{b}_0$ *in the basis has the following property:*

$$\|\mathbf{b}_0\|_2 = \delta_0^n \operatorname{vol}(L)^{1/n} .$$

We may also refer to δ_0 *itself, and call it the* root-Hermite factor*. The logarithm to base 2 of* δ_0 *is called the* log root-Hermite factor*.*

The Gram-Schmidt vectors of a reduced lattice basis are commonly estimated using the *Geometric Series Assumption* [201].

Heuristic 2. *Let* $\{\mathbf{b}_0 \dots \mathbf{b}_{n-1}\}$ *be a basis of a lattice* L *that is output by a lattice basis reduction algorithm of quality given by root-Hermite factor* δ_0*. Let* $\mathbf{b}_i^*$ *denote the corresponding Gram-Schmidt vectors. Then the* Geometric Series Assumption *asserts that, for some* $0 < \upsilon < 1$*, the length of* $\mathbf{b}_i^*$ *is approximated by*

$$\|\mathbf{b}_i^*\|_2 \approx \upsilon^i \, \|\mathbf{b}_0\|_2 \, .$$

Using Heuristic 2 we have that

$$\mathrm{vol}\,(L)^{\frac{1}{n}} = \prod_{i=0}^{n-1} \|\mathbf{b}_i^*\|_2 = \left(\prod_{i=0}^{n-1} \upsilon^i \, \|\mathbf{b}_0\|_2 \right)^{\frac{1}{n}} = \sqrt{\|\mathbf{b}_0\|_2^2 \, \upsilon^{n-1}} = \|\mathbf{b}_0\|_2 \, \sqrt{\upsilon^{n-1}} \, .$$

Using the Definition 12, we can conclude that $\delta_0^{-n} = \sqrt{\upsilon^{n-1}}$ and hence $\upsilon = \delta_0^{-2n/(n+1)}$. Approximating this as $\upsilon \approx \delta_0^{-2}$ and again using Definition 12 gives a reformulation of Heuristic 2 as

$$\|\mathbf{b}_i\|_2^* \approx \delta_0^{-2i+n} \, \mathrm{vol}\,(L)^{\frac{1}{n}} \, .$$

2.3 The Learning with Errors problem

The Learning with Errors (LWE) problem was introduced by Regev [194, 195] and is provably as hard as certain worst-case lattice problems [194, 45]. LWE can be thought of as a generalisation of the Learning Parity with Noise (LPN) problem [137] into large moduli q. We now define LWE and the related Short Integer Solutions problem [5, 167].

Definition 13. *The* Learning with Errors *problem (LWE) is defined as follows. Let* n*,* q *be positive integers,* χ *be a probability distribution on* $\mathbb{Z}$ *and* $\mathbf{s}$ *be a secret vector in* $\mathbb{Z}_q^n$*. We denote by* $L_{\mathbf{s},\chi}$ *the probability distribution on* $\mathbb{Z}_q^n \times \mathbb{Z}_q$ *obtained by choosing* $\mathbf{a} \in \mathbb{Z}_q^n$ *uniformly at random, choosing* $e \in \mathbb{Z}$ *according to* χ *and considering it in* $\mathbb{Z}_q$ *and returning*

$$(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q \, .$$

Decision-LWE *is the problem of deciding whether pairs* $(\mathbf{a}, c) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *are sampled according to* $L_{\mathbf{s},\chi}$ *or the uniform distribution on* $\mathbb{Z}_q^n \times \mathbb{Z}_q$*.*

Search-LWE *is the problem of recovering* $\mathbf{s}$ *from* $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *sampled according to* $L_{\mathbf{s},\chi}$.

Definition 14. *The* Short Integers Solutions *problem (SIS) is defined as follows. Let* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *be a matrix. Let* $\beta \in \mathbb{R}$ *be a bound. Find a short vector* $\|\mathbf{v}\|_2 < \beta$ *in the lattice*

$$L = \{\mathbf{w} \in \mathbb{Z}_q^m \mid \mathbf{w}\mathbf{A} \equiv 0 \mod q\}.$$

Typically, the distribution χ in $L_{\mathbf{s},\chi}$ is a discrete Gaussian distribution on $\mathbb{Z}$ with *centre parameter* zero and *width parameter* αq, denoted by $\mathcal{D}_{\mathbb{Z},\alpha q}$. A discrete Gaussian distribution with centre parameter μ and width parameter αq samples elements with a probability proportional to $\exp(-\pi \frac{(x-\mu)^2}{(\alpha q)^2})$. The standard deviation of a continuous Gaussian with width parameter αq is $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$.

The *smoothing parameter* $\eta_\epsilon(L)$ of a lattice L [167] is equal to the smallest $s > 0$ such that the distribution obtained by sampling a continuous Gaussian of width parameter s and reducing modulo L is very close to uniform. In particular, the two distributions have a pointwise probability density to within a $(1 \pm \epsilon)$ factor of each other [74]. If $\sigma \geq \eta_\epsilon(\mathbb{Z})$ is the standard deviation of a discrete Gaussian with width parameter αq, then we also have that $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$ is a good approximation [96].

We will use the following standard fact about the Gaussian distribution.

Lemma 3. *Let* χ *denote the Gaussian distribution with standard deviation* σ*. For* $x > 0$*, denote* $Q(x) = \left(1 - \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$*. Then, for all* $C > 0$*,*

$$\mathbf{P}(e \xleftarrow{\$} \chi : |e| > C \cdot \sigma) \approx Q(C) \leq \frac{2e^{-C^2/2}}{C\sqrt{2\pi}}.$$

Suppose we sample m times independently from the distribution $L_{\mathbf{s},\chi}$. The resulting LWE instance is parameterised by its dimension n, its modulus q, the number of samples m and the error distribution, characterised by the parameter α. This gives the most generic characterisation of an LWE instance.

In this thesis we are mainly concerned with choosing parameters for security and hence a key question is how hard it is to solve a given LWE instance. To consider the widest range of possible approaches, we typically assume the attacker, who is trying

to solve LWE, may query $L_{\mathbf{s},\chi}$ for as many samples m as they like. One argument why this is reasonable is that given a fixed polynomial number of samples, we can generate arbitrarily many more independent samples, with only a slight worsening in the error [111, 23, 196]. In addition, there is typically an optimal choice for m, and the best strategy if the number of available samples is larger is to discard the extra samples. Therefore, in most cases we will ignore m, or assume it is chosen to be the most convenient value, and characterise LWE instances just by n, α, and q.

A typical choice for q and α is given by the following. Inspired by Regev's choice [195] of $\alpha q = 2\sqrt{n}$, we let $c > \frac{1}{2}$ be a small constant, then choose $q = n^c$ and $\alpha \approx n^{\frac{1}{2}-c}$. In this case we may characterise the LWE instance by n and c. This choice of width parameter $\alpha q \approx \sqrt{n}$ can be considered typical since we require $\alpha q > \sqrt{n}$ for the reduction from GapSVP to LWE to go through [195] and furthermore if $\alpha q < \sqrt{n}$ then Arora and Ge's algorithm [25] becomes subexponential.

In some applications, the secret $\mathbf{s}$ does not have components $\mathbf{s}_{(i)}$ chosen from the whole of $\mathbb{Z}_q$ as in Definition 13 but instead chosen such that they are small, for example taking values in $\{0,1\}$ or $\{-1,0,1\}$. We call such an LWE instance a *small secret* LWE instance. In this case we characterise the instance by n, α, q, and ψ where ψ is the distribution of the $\mathbf{s}_{(i)}$.

In Chapter 5 we will consider a variant of LWE where χ is not a discrete Gaussian but instead is the uniform distribution on $\{0,1\}$. We call this LWE with binary error.

Definition 15. *The* LWE with binary error *problem is defined as follows. Let n, q be positive integers, $\mathcal{U}$ be the uniform distribution on $\{0,1\}$ and $\mathbf{s} \xleftarrow{\$} \mathcal{U}^n$ be a secret vector in $\{0,1\}^n$. We denote by $L_{\mathbf{s},\mathcal{U}}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \xleftarrow{\$} \mathcal{U}$ and returning*

$$(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q \,.$$

The search variant of LWE with binary error is the problem of recovering $\mathbf{s}$ from m samples $(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s}_i \rangle + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ sampled according to $L_{\mathbf{s},\mathcal{U}}$, with $i \in \{1, \ldots, m\}$. The decision variant of LWE with binary error is the problem of deciding whether m pairs $(\mathbf{a}_i, c_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $i \in \{1, \ldots, m\}$ are sampled according to $L_{\mathbf{s},\mathcal{U}}$ or the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

We show in Lemma 4 that we can transform an LWE instance where the secret is chosen uniformly at random from $\mathbb{Z}_q^n$ into one where the secret vector follows the same distribution as the error, at the cost of n samples. Lemma 4 considers LWE with a discrete Gaussian error distribution, but the same argument can be generalised to other error distributions, for example a uniform binary error distribution. This is why in Definition 15 we used a secret following the same distribution as the error, rather than a general secret, chosen uniformly at random from $\mathbb{Z}_q^n$.

Lemma 4 ([23])**.** *Let q be prime. Let $\mathcal{D}_{\mathbb{Z}^n,\alpha q}$ be an n-dimensional extension of $\mathcal{D}_{\mathbb{Z},\alpha q}$ where each component is sampled according to $\mathcal{D}_{\mathbb{Z},\alpha q}$. Then, given access to an oracle $L_{\mathbf{s},\chi}$ returning samples of the form $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $\mathbf{a} \xleftarrow{\$} \mathcal{U}\left(\mathbb{Z}_q^n\right)$, $e \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\alpha q}$ and $\mathbf{s} \in \mathbb{Z}_q^n$, we can construct samples of the form*

$$(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{e}\rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$$

with $\mathbf{a} \xleftarrow{\$} \mathcal{U}\left(\mathbb{Z}_q^n\right)$, $e \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\alpha q}$ and $\mathbf{e} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^n,\alpha q}$ in $2\,n^2$ operations in $\mathbb{Z}_q$ per sample, at the loss of n samples overall and with $\mathcal{O}\left(n^\omega\right)$ operations for precomputation.

Proof: Take n samples from $L_{\mathbf{s},\chi}$ and write:

$$(\mathbf{A}_0, \mathbf{c}_0) = (\mathbf{A}_0, \mathbf{A}_0 \cdot \mathbf{s} + \mathbf{e}_0)$$

where $\mathbf{A}_0 \in \mathbb{Z}_q^{n\times n}$. With probability $\prod_{i=1}^{n}(q^n - q^{i-1})/q^{n^2}$ this matrix is invertible. Precompute $\mathbf{A}_0^{-1}$ and store it; this costs $\mathcal{O}\left(n^\omega\right)$ operations. Now, to produce samples of the form $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{e}\rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $\mathbf{a} \xleftarrow{\$} \mathcal{U}\left(\mathbb{Z}_q^n\right)$, $e \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\alpha q}$ and $\mathbf{e} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^n,\alpha q}$ we sample

$$(\mathbf{a}_1, c_1) = (\mathbf{a}_1, \langle \mathbf{a}_1, \mathbf{s}\rangle + e_1)$$

from $L_{\mathbf{s},\chi}$ and compute:

$$\begin{aligned}
\mathbf{a}_1 \cdot \mathbf{A}_0^{-1} \cdot \mathbf{c}_0 - c_1 &= \mathbf{a}_1 \cdot \mathbf{A}_0^{-1}(\mathbf{A}_0 \cdot \mathbf{s} + \mathbf{e}_0) - \mathbf{a}_1 \cdot \mathbf{s} - e_1 \\
&= \mathbf{a}_1 \cdot \mathbf{A}_0^{-1} \cdot \mathbf{A}_0 \cdot \mathbf{s} + \mathbf{a}_1 \cdot \mathbf{A}_0^{-1}\mathbf{e}_0 - \mathbf{a}_1 \cdot \mathbf{s} - e_1 \\
&= \mathbf{a}_1 \cdot \mathbf{s} + \mathbf{a}_1 \cdot \mathbf{A}_0^{-1}\mathbf{e}_0 - \mathbf{a}_1 \cdot \mathbf{s} - e_1 \\
&= \mathbf{a}_1 \cdot \mathbf{A}_0^{-1}\mathbf{e}_0 - e_1\,.
\end{aligned}$$

Since $\mathcal{D}_{\mathbb{Z},\alpha q}$ is symmetric and $\mathbf{A}_0^{-1}$ has full rank, we see that

$$(\mathbf{a}_1 \cdot \mathbf{A}_0^{-1}, \mathbf{a}_1 \cdot \mathbf{A}_0^{-1}\mathbf{c}_0 + c_1) = (\mathbf{a}_1 \cdot \mathbf{A}_0^{-1}, \mathbf{a}_1 \cdot \mathbf{A}_0^{-1}\mathbf{e}_0 - e_1)$$

is a valid sample of the form $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{e} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $\mathbf{a} \xleftarrow{\$} \mathcal{U}\left(\mathbb{Z}_q^n\right)$, $e \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\alpha q}$ and $\mathbf{e} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^n,\alpha q}$. □

We now show that Decision-LWE and Search-LWE are equivalent. The direction from search to decision is trivial: suppose samples $(\mathbf{a}_i, c_i)$ are given to a search oracle, and it returns a vector $\mathbf{s}$. We then compute $c_i - \langle \mathbf{a}_i, \mathbf{s} \rangle$, which is distributed according to χ if $(\mathbf{a}_i, c_i)$ are LWE samples, and is uniformly distributed if $(\mathbf{a}_i, c_i)$ is uniform. Hence we can distinguish these cases. The nontrivial direction, due to Regev [195], is reproduced in Lemma 5. Having established equivalence, whenever a method can be shown to solve Search-LWE or Decision-LWE we can speak of it solving LWE.

Lemma 5 ([195])**.** *Let $n \geq 1$ be an integer, $2 \leq q \leq \text{poly}(n)$ be a prime, and χ be some distribution on $\mathbb{Z}_q$. Assume that we have access to a procedure W that, for all $\mathbf{s}$, accepts with probability exponentially close to 1 on inputs from $L_{\mathbf{s},\chi}$ and rejects with probability exponentially close to 1 on uniformly random inputs. Then, there exists an efficient algorithm W' that, given samples from $L_{\mathbf{s},\chi}$ for some $\mathbf{s}$, outputs $\mathbf{s}$ with probability exponentially close to 1.*

Proof: We show how W' finds the first component $\mathbf{s}_{(0)}$ of $\mathbf{s}$; finding the other components is similar. For any $k \in \mathbb{Z}_q$ consider the following transformation. Given a pair $(\mathbf{a}, c)$ as input to W', let it output the pair $(\mathbf{a} + (l, 0, \ldots, 0), c + lk)$ where $l \in \mathbb{Z}_q$ is chosen uniformly at random. It is easy to see that this transformation takes the uniform distribution to itself. On the other hand suppose the input pair $(\mathbf{a}, c)$ is sampled from $L_{\mathbf{s},\chi}$. If $k = \mathbf{s}_{(0)}$ then this transformation takes $L_{\mathbf{s},\chi}$ into itself. If $k \neq \mathbf{s}_{(0)}$ then this transformation takes $L_{\mathbf{s},\chi}$ to the uniform distribution. There are only polynomially many (namely q) possibilities for $\mathbf{s}_{(0)}$, so we can try all of them as possible k values. For each k value, let the output of W' be the input to W. Then as W can distinguish $L_{\mathbf{s},\chi}$ from uniform, it can tell whether $k = \mathbf{s}_{(0)}$. □

In certain cases, given samples from $L_{\mathbf{s},\chi}$, we can construct LWE instances where the modulus is now p for some particular $p < q$ using a technique known as *modulus switching*. This was initially introduced to improve homomorphic encryption [46] but, as we will discuss in Chapter 5, can also be employed to reduce the cost of solving LWE in certain cases. In the context of solving LWE, modulus switching is the process of considering an instance of LWE with modulus q as a scaled instance

of LWE with modulus p. Modulus switching is achieved by scaling and rounding, and this process incurs a noise increase that depends on $\mathbf{s}$. Therefore, the technique can only be used for LWE instances with sufficiently small secret.

Lemma 6 ([46, 45]). *Let $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ be sampled from $L_{\mathbf{s},\chi}$. Let $p \approx \sqrt{\frac{2\pi n}{12}} \cdot \frac{\sigma_s}{\alpha}$, where σ_s is the standard deviation of elements in the secret $\mathbf{s}$. If $p < q$ then $\left(\left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \left\lfloor \frac{p}{q} \cdot c \right\rceil \right) \in \mathbb{Z}_p^n \times \mathbb{Z}_p$ follows a distribution close to $L_{\mathbf{s},\mathcal{D}_{\mathbb{Z},\sqrt{2}\alpha p + \mathcal{O}(1)}}$. Hence, we can heuristically treat $\left(\left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \left\lfloor \frac{p}{q} \cdot c \right\rceil \right)$ as an LWE sample parameterised by n, $\sqrt{2}\alpha$, p.*

Proof: Consider

$$\begin{aligned}
\left\lfloor \frac{p}{q} \cdot c \right\rceil &= \left\lfloor \frac{p}{q} \left(\langle \mathbf{a}, \mathbf{s} \rangle + e \right) \right\rceil \\
&= \left\lfloor \left\langle \frac{p}{q} \cdot \mathbf{a}, \mathbf{s} \right\rangle_p + \frac{p}{q} \cdot e \right\rceil \\
&= \left\lfloor \left\langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p + \left\langle \frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p + \frac{p}{q} \cdot e \right\rceil \\
&= \left\langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p + \left\langle \frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p + \frac{p}{q} \cdot e + e' \\
&= \left\langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p + e'' + \frac{p}{q} \cdot e + e' ,
\end{aligned}$$

where $e' \in [-0.5, 0.5]$ is the error from rounding and $e'' = \left\langle \frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p$. As an inner product, e'' approaches a discrete Gaussian as n increases. Since the rounding $\frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil$ takes values uniformly in $[-0.5, 0.5]$ we have that $e'' \approx \sqrt{n/12}\,\sigma_s$. We have that $\frac{p}{q} \cdot e$ is also a discrete Gaussian, since it is a scaling of e. We choose p to balance the dominant noise terms and thus the resulting overall noise is approximated by a discrete Gaussian (plus a small $\mathcal{O}(1)$ noise given by the e' term). We require

$$\begin{aligned}
\frac{p}{q} \cdot \alpha q / \sqrt{2\pi} &\approx \sqrt{n/12}\,\sigma_s \\
p \cdot \alpha / \sqrt{2\pi} &\approx \sqrt{n/12}\,\sigma_s \\
p &\approx \frac{\sqrt{2\pi}}{\alpha} \cdot \sqrt{n/12}\,\sigma_s \\
p &\approx \sqrt{\frac{2\pi n}{12}} \cdot \frac{\sigma_s}{\alpha} .
\end{aligned}$$

□

We note that $\left(\left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \left\lfloor \frac{p}{q} \cdot c \right\rceil\right)$ can only heuristically be considered as an LWE sample parameterised by n, $\sqrt{2}\alpha$, p because $\left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil$ is not exactly uniform and the error distribution is not exactly $\mathcal{D}_{\mathbb{Z},\sqrt{2}\alpha p}$. This is not problematic from a cryptanalytic point of view because, as we will see in Chapter 4, the complexity of attacks typically depends on the size of the error, rather than its exact distribution. For BKW-type algorithms (see Section 4.5), the non-uniformity of $\left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil$ may even be beneficial as we would expect to require fewer samples before encountering a pair that agrees on the first several components.

In Lemma 6, we chose p to ensure that $\left\| \left\langle \frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle_p \right\|_2 \approx \frac{p}{q} \cdot \|e\|_2$ if $\mathbf{s}$ is small enough. This means that after modulus switching, the new error $e'' + \frac{p}{q} \cdot e + e'$ is essentially a scaling of the previous error e (ignoring the small contribution of e'). In principle, one can modulus switch to any p, even one larger than q, but reducing p is typically advantageous as it will reduce the running time of most algorithms. However, picking p too small could make e' or e'' a more dominant noise term and increase the overall noise level leading to a higher solving complexity.

Chapter 3

Lattice reduction

Contents

Many algorithms for solving LWE rely on a lattice reduction step. In this chapter we review lattice reduction algorithms and discuss estimates in the literature for their running time. We also give a new result on the asymptotic time complexity of BKZ. This chapter is based on material presented in [16].

As discussed in Chapter 2, in dimension $n \geq 2$ there are infinitely many bases of a lattice. A natural question is whether one possible basis should be preferred to another possible basis. Indeed, a basis that would be considered good is one that has short vectors that are close to orthogonal. Knowing a good basis typically makes lattice problems, such as finding the lattice point closest to a given target point, easier to solve. The goal of a lattice basis reduction algorithm is to transform an input lattice basis into a better basis, with vectors that are shorter and more orthogonal. Recall that the quality of a basis that is output by a lattice reduction algorithm is characterised by the Hermite factor δ_0^n, which is easy to compute.

BKZ [202] is a prominent family of lattice reduction algorithms, parameterised by a block size k, that we will discuss in detail. When $k = 2$ the algorithm produces an output basis containing a short vector that is only guaranteed to be within an exponential factor of a shortest vector. In particular, the basis that is output sat-

isfies the *LLL-reduced* property of Lenstra, Lenstra and Lovász [148], specified in Definition 16. The definition depends on a value $\delta \in (\frac{1}{4}, 1)$; a typical choice for this value is $\delta = \frac{3}{4}$.

Definition 16 ([148])**.** *Let* $\{\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}\}$ *be a basis of a lattice. Let* $\mathbf{b}_i^*$ *be the Gram-Schmidt vectors and* $\mu_{i,j}$ *be the Gram-Schmidt coefficients. The basis is* size reduced *if* $|\mu_{i,j}| \leq \frac{1}{2}$ *for* $0 \leq j < i \leq n-1$*. Let* $\delta \in (\frac{1}{4}, 1)$*. The* Lovász condition *is* $\frac{\|\mathbf{b}_i^*\|_2}{\|\mathbf{b}_{i-1}^*\|_2} \geq \delta - \mu_{i,i-1}^2$*. The basis is* LLL-reduced *if it is size reduced and satisfies the Lovász condition for some* δ*.*

When the block size $k = n$; that is, the block size is equal to the full size of the basis, then the output basis is said to be *HKZ-reduced*, after Hermite, Korkine and Zolotarev [127, 140]. Let $\mathbf{v}_0, \ldots, \mathbf{v}_{m-1}$ be vectors spanning a subspace V. The orthogonal complement $V^\perp = \langle \mathbf{v}_0, \ldots, \mathbf{v}_{m-1} \rangle^\perp$ is the set of all vectors orthogonal to every vector in V.

Definition 17. *Let* $\{\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}\}$ *be a basis of a lattice* L *and let* $\{\mathbf{b}_0^*, \ldots, \mathbf{b}_{n-1}^*\}$ *be the Gram-Schmidt vectors. The basis is* HKZ-reduced *if* $\|\mathbf{b}_0\|_2 = \lambda_1(L)$ *and* $\|\mathbf{b}_i^*\|_2 = \lambda_1(\pi_i(L))$ *where* $\pi_i(L) = \langle \mathbf{b}_0, \ldots, \mathbf{b}_{i-1} \rangle^\perp$ *for* $1 \leq i \leq n-1$*.*

In some sense, a HKZ-reduced basis is optimally reduced, since it contains a shortest nonzero vector of the lattice. Indeed, each HKZ-reduced basis vector is known to be close in length to the respective successive minimum.

Theorem 1 ([144, 161])**.** *Let* $\{\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}\}$ *be a HKZ-reduced basis of a lattice* L*. Then for* $0 \leq i \leq n-1$*,*

$$\frac{4}{i+4} \leq \frac{\|\mathbf{b}_i\|_2^2}{\lambda_{i+1}^2(L)} \leq \frac{i+4}{4}\,.$$

However, BKZ with block size n runs in at least exponential time in the worst case. In practice, a variant of BKZ with some intermediary block size $2 < k < n$ would be used.

In the remainder of this chapter we review the lattice reduction algorithms LLL [148] and BKZ [202]. For brevity we do not describe Slide reduction [103] or Dual BKZ [171], although Micciancio and Walter [171] show experimentally that these are competitive with BKZ: for example, the average root-Hermite factor achieved by these

algorithms is shown to converge to that of BKZ. We introduce LLL in Section 3.1 and BKZ in Section 3.2. We conclude in Section 3.3 by specifying the runtime estimates for lattice reduction that will be used later in the thesis.

3.1 The LLL algorithm

Introduced by Lenstra, Lenstra and Lovász in [148], the LLL algorithm was initially proposed as an algorithm for factoring polynomials, but has since found a wealth of applications [179]. The LLL algorithm can be considered as a generalisation of the two dimensional lattice reduction algorithm due to Lagrange (see, for example, [134, 179]), which is sometimes attributed to Gauss.

The output of Lagrange's lattice reduction algorithm is a basis $\{\mathbf{b}_0, \mathbf{b}_1\}$ such that $\lambda_1 = \|\mathbf{b}_0\|_2 \leq \|\mathbf{b}_1\|_2 = \lambda_2$ and the Gram-Schmidt coefficient $|\mu_{1,0}| \leq \frac{1}{2}$. The algorithm works by taking in a pair of vectors $\{\mathbf{b}_0, \mathbf{b}_1\}$ arranged such that $\|\mathbf{b}_0\|_2 \leq \|\mathbf{b}_1\|_2$ and then setting $\mathbf{b}_1 = \mathbf{b}_1 - \lfloor \mu_{1,0} \rceil \mathbf{b}_0$, then swapping the vectors and repeating until no more changes can be made. Thus, when this terminates, we must have $|\mu_{1,0}| \leq \frac{1}{2}$. To extend into higher dimensions we would like to do something similar, but the optimal way to do this is not immediately clear because of the additional choice of directions. As we will see below, the LLL algorithm can be seen as a generalisation of the Lagrange algorithm that is relaxed in some sense; other generalisations are discussed for example in [178].

The Lagrange algorithm ensures that $\frac{\|\mathbf{b}_1\|_2}{\|\mathbf{b}_0\|_2}$ is not too small; in particular, we have $\frac{\|\mathbf{b}_1\|_2}{\|\mathbf{b}_0\|_2} \geq 1 - \mu_{1,0}^2$. The Lovász condition (recall Definition 16) is a relaxed general version of this. Essentially, the LLL algorithm works by size reducing each basis vector, updating the Gram-Schmidt vectors and coefficients, and then checking if the Lovász condition still holds; if it does not, then it swaps the current vector with the previous vector. In more detail, let the input basis be $\{\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}\}$. Starting at $i = 1$ and incrementing upwards, consider $\mathbf{b}_i$ and size reduce with respect to $\mathbf{b}_j$ for $j = i - 1$ down to $j = 0$. Then check if $\mathbf{b}_i$ and $\mathbf{b}_{i-1}$ satisfy the Lovász condition. If they do, increment i; if not, swap them and decrement i to ensure the swap has not affected the Lovász condition holding in the previous pair.

Lenstra, Lenstra and Lovász in [148] showed that their algorithm runs in polynomial time. Several provable variants of the LLL algorithm, which improve the running time, have been proposed [135, 200, 176, 181, 175]. Suppose a bound B is known on the norms of the input basis, so that $\|\mathbf{b}_i\|_2 < B$ for all $0 \leq i \leq n-1$. The variant due to Neumaier and Stehlé [175], for example, runs in time $\mathcal{O}\left(n^{4+\epsilon} \log^{1+\epsilon} B\right)$ for any $\epsilon > 0$. Heuristically, variants of LLL, such as L^2 [176], perform even better: for example, Stehlé [206] showed that $\mathcal{O}\left(n^2 \log^2 B\right)$ can be achieved in practice.

In terms of quality of output, the LLL algorithm theoretically achieves a Hermite factor of $\left(\frac{4}{3}\right)^{\frac{n-1}{4}}$ [148], although in practice, the output quality is much better. Nguyen and Stehlé [177] conclude from experiments in dimension up to $n = 130$ that the root-Hermite factor δ_0 appears to converge towards 1.02 as the dimension increases. For comparison, the theoretical root-Hermite factor in dimension 130 is approximately 1.074.

3.2 The BKZ algorithm

The BKZ algorithm was introduced by Schnorr and Euchner [202]. As mentioned above, BKZ is in fact a family of algorithms parameterised by a block size k. We call a basis that is output by the BKZ algorithm with block size k a *BKZ_k-reduced* basis. The BKZ algorithm requires an algorithm solving exact SVP in dimension up to k as a subroutine, which we refer to as calling an SVP oracle.

The BKZ algorithm with block size k runs as follows, where at every stage the updated basis is denoted $\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}$. The input basis is LLL-reduced, and the first block is $\mathbf{b}_0, \ldots, \mathbf{b}_{k-1}$. We call the SVP oracle to obtain a short vector, $\mathbf{b}'_0$, in the space spanned by these vectors. We now have $k+1$ vectors spanning a k dimensional space, so we call LLL to obtain a new set of k linearly independent vectors. The second block is made of vectors that are the projections of $\mathbf{b}_1, \ldots, \mathbf{b}_k$ onto $\langle \mathbf{b}_0 \rangle^\perp$ (the space that is the span of the orthogonal complement of $\mathbf{b}_0$). Again we call the SVP oracle to obtain a short vector in this space, $\mathbf{b}'_1$, which can be viewed as the projection of some $\mathbf{b}''_1$ in the lattice. Now we call LLL on $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_k, \mathbf{b}''_1$ to update the list of basis vectors. The next block is made of vectors that are the projection of $\mathbf{b}_2, \ldots, \mathbf{b}_{k+1}$ onto $\langle \mathbf{b}_0, \mathbf{b}_1 \rangle^\perp$ (the space that is the span of the orthogonal complement

of $\mathbf{b}_0$ and $\mathbf{b}_1$), and again the SVP oracle is called to obtain a short vector in this space, which can be viewed as a projected $\mathbf{b}_2''$; and this procedure carries on through the basis. The first $n-k+1$ blocks are all of size k, and then after this point each block is one vector shorter than the previous block. The output basis of this process is another LLL-reduced basis, which can be treated as a new input, and the whole process continues, until a basis passes through unchanged, at which point the algorithm terminates. We can see that BKZ constructively achieves a basis with the following property: each block of size k, that is, all the first $n-k+1$ blocks, is a HKZ-reduced basis.

To implement the SVP oracle in dimension k, four main possibilities exist: sieving, enumeration, computing the Voronoi cell and Discrete Gaussian sampling. Discrete Gaussian sampling takes $2^{k+o(k)}$ operations and $2^{k+o(k)}$ memory [4]. Computing the Voronoi cell of the lattice takes $2^{2k+o(k)}$ operations and $2^{k+o(k)}$ memory [169]. The provable variant of sieving takes $2^{k+o(k)}$ operations and $2^{k+o(k)}$ memory [192, 4]. The best heuristic classical variant of sieving takes $2^{0.292\,k+o(k)}$ operations [33] and the best heuristic quantum variant of sieving takes $2^{0.265\,k+o(k)}$ operations [141]. The running time of provable variants of enumeration depends on the preprocessing. Fincke and Pohst [101] show that by preprocessing with LLL, enumeration requires $2^{\mathcal{O}(k^2)}$ operations and poly(k) memory. Kannan [136] showed that with stronger preprocessing, enumeration be done in $k^{\mathcal{O}(k)}$ operations and poly(k) memory. Achieving $k^{\mathcal{O}(k)}$ was considered prohibitively expensive in practice until recently, but Micciancio and Walter [170] proposed a variant that achieves $k^{\mathcal{O}(k)}$ with smaller overhead. Moreover, Walter [213] showed that preprocessing local blocks with BKZ-k' where $k' = \mathcal{O}(k)$ before enumeration also reduces the complexity of BKZ-k to $k^{\mathcal{O}(k)}$.

In terms of quality of output, Chen [64] gives a limiting value of the root-Hermite factor δ_0 achievable by BKZ as a function of the block size k. Experimental evidence suggests that we may use Chen's limiting value as an estimate also when n is finite.

Lemma 7 ([64])**.** *Let v_k be the volume of the unit ball in dimension k and let δ_0 be the root-Hermite factor achievable by BKZ with block size k. Assume that both Heuristic 1 and Heuristic 2 hold. Then*

$$\lim_{n\to\infty} \delta_0 = \left(v_k^{\frac{-1}{k}}\right)^{\frac{1}{k-1}} \approx \left(\frac{k}{2\pi e}(\pi k)^{\frac{1}{k}}\right)^{\frac{1}{2(k-1)}}.$$

Another approximation for the δ_0 achievable for a fixed k is known as the *lattice rule of thumb* (see, for example [207]) and states that $\delta_0 = k^{\frac{1}{2k}}$. This expression itself is often [208] approximated by $\delta_0 = 2^{\frac{1}{k}}$. However, we will show that this simplification implies a subexponential algorithm for solving LWE.

Several possible improvements to BKZ have been suggested, namely early termination [124], local block preprocessing, extreme pruning [105] and optimising the enumeration radius. These improvements have been combined and tested as so-called BKZ 2.0 [65].

No closed formula for the expected number of BKZ rounds ρ is known and the best upper bound is exponential [124, 104]. However, it has been observed that the quality of the output basis increases more dramatically in the earlier rounds of BKZ, and this gives rise to the idea of early termination. Assuming n calls to an SVP oracle per round, Hanrot *et al.* [124] show that after $\rho = \frac{n^2}{k^2} \log n$ many rounds, the quality of the basis is already very close to the final output. The choice to terminate BKZ early may therefore still return a basis close to the desired quality, whereas performing more rounds would increase the runtime while not increasing the basis quality much.

Recall that in BKZ the local basis (of a block) is always LLL-reduced. Local block preprocessing involves performing a stronger reduction of the local basis, so that it is more than just LLL-reduced. For example, it could be a basis that is $\text{BKZ}_{k'}$-reduced with some block size $k' < k$. Chen and Nguyen [65] use a recursive BKZ with early termination as preprocessing. We can also consider a global preprocessing: computing a BKZ_k-reduced basis by beginning with a $\text{BKZ}_{k'}$-reduced basis for some $k' < k$, rather than an LLL-reduced basis. The $\text{BKZ}_{k'}$-reduced basis could itself be computed by beginning with a $\text{BKZ}_{k''}$-reduced basis for some $k'' < k'$, rather than an LLL-reduced basis, and so on. This idea is known as *progressive BKZ* [22].

Pruning is an improvement that can be applied when the SVP oracle is implemented as enumeration. Rather than exploring all branches in the search tree, we instead explore only a subset of the branches, according to some rule. This is faster than exploring all branches, but may not return the shortest vector.

For optimising the enumeration radius, Chen and Nguyen [65] use $\nu = \sqrt{1.1}$ times the Gaussian Heuristic as an upper bound for the enumeration radius, where the fixed value ν is determined from experiments, while Aono *et al.* [22] allow the value ν to be varied.

3.2.1 Asymptotic running time of BKZ

In this section we consider the asymptotic runtime behaviour of the BKZ algorithm, and in the following section we will discuss estimates from in the literature for the running time of BKZ in practice.

We determine an asymptotic lower bound of the running time of BKZ required to achieve a certain basis quality characterised by δ_0 as a function of δ_0 as follows. We know that BKZ with block size k should cost at least as much as the cost of calling an SVP oracle in dimension k. Thus, we can write k in terms of δ_0, using one of the estimates of quality of output of BKZ with block size k. We then substitute this value for k into an appropriate asymptotic cost of an SVP oracle.

In order to relate k and δ_0 we use the lattice rule of thumb (recall Section 3.2). Recall that this states $\delta_0 = k^{\frac{1}{2k}}$, which implies $\frac{k}{\log k} = \frac{1}{2 \log \delta_0}$. We will lower bound k using Lemma 8.

Lemma 8. *Let $g_n(t)$ for some t be the sequence defined such that $g_0(t) = 2$ and for $i \geq 1$, $g_i(t) = t \log(g_{i-1}(t))$. For $a > 0$, define $b = \frac{a}{\log a}$. If $a \geq 2$ then $a \geq g_n(b)$ for any $n \geq 0$.*

Proof: We prove the claim by induction. For the base case, by assumption we have $a \geq 2 = g_0(b)$. For the inductive step, suppose $a \geq g_i(b)$. This implies

$$\begin{aligned} \log a &\geq \log\left(g_i(b)\right) \\ b \log a &\geq b \log\left(g_i(b)\right) \\ b \log a &\geq g_{i+1}(b) \\ a &\geq g_{i+1}(b) \end{aligned}$$

since $b \log a = a$ by definition, and we are done. □

In BKZ the block size $k \geq 2$, so by Lemma 8 we have $k \geq g_n\left(\frac{1}{2\log\delta_0}\right)$ for all n. In particular, we can use $k \geq g_2\left(\frac{1}{2\log\delta_0}\right) = \frac{-\log(2\log\delta_0)}{2\log\delta_0}$ to lower bound the log of the asymptotic time complexity of the BKZ algorithm as follows.

Corollary 1. *The log of the asymptotic time complexity for running BKZ to achieve a root-Hermite factor δ_0 is:*

$$\begin{aligned}
\Omega\left(\frac{\log^2(2\log\delta_0)}{4\log^2\delta_0}\right) &\qquad \text{if calling the SVP oracle costs } 2^{\mathcal{O}(k^2)},\\
\Omega\left(\frac{-\log\left(\frac{-\log(2\log\delta_0)}{2\log\delta_0}\right)\log(2\log\delta_0)}{2\log\delta_0}\right) &\qquad \text{if calling the SVP oracle costs } k^{\mathcal{O}(k)},\\
\Omega\left(\frac{-\log(2\log\delta_0)}{2\log\delta_0}\right) &\qquad \text{if calling the SVP oracle costs } 2^{\mathcal{O}(k)}.
\end{aligned}$$

For sufficiently large k, the bound of Lemma 8 becomes tighter as n increases. In particular, applying the following lemma shows that k is the limit of the sequence $g_n\left(\frac{1}{2\log\delta_0}\right)$ when $k \geq 4$.

Lemma 9. *Let $g_n(t)$ for some t be the sequence defined such that $g_0(t) = 2$ and for $i \geq 1$, $g_i(t) = t\log(g_{i-1}(t))$. Define $g_\infty(t) = \lim_{n\to\infty} g_n(t)$, if it exists. For $a > 0$, define $b = \frac{a}{\log a}$. If $a \geq 4$, then $a = g_\infty(b)$.*

Proof: By Lemma 8, since $a \geq 4 > 2$, the sequence $g_n(b)$ is bounded above by a. We will show that $g_n(b)$ is a monotonic sequence and hence we can conclude it is convergent. We will use the fact that if $a \geq 4$ then $\frac{a}{\log a} = b \geq 2$. We prove by induction that $g_n(b) \geq g_{n-1}(b)$ for all $n \geq 1$. For the base case, we have

$$g_1(b) = b\log(g_0(b)) = b\log 2 = b \geq 2 = g_0(b)\,.$$

For the inductive step, suppose $g_i(b) \geq g_{i-1}(b)$. This implies

$$\begin{aligned}
\frac{g_i(b)}{g_{i-1}(b)} &\geq 1\\
\log\left(\frac{g_i(b)}{g_{i-1}(b)}\right) &\geq 0\\
b\log\left(\frac{g_i(b)}{g_{i-1}(b)}\right) &\geq 0\\
b\log g_i(b) - b\log g_{i-1}(b) &\geq 0\\
g_{i+1}(b) - g_i(b) &\geq 0\\
g_{i+1}(b) &\geq g_i(b)
\end{aligned}$$

as required. We have shown than $g_n(b)$ is convergent and we may denote its limit by $g_\infty(b)$. This satisfies $g_\infty(b) = b \log(g_\infty(b))$ and since $b = \frac{a}{\log a}$ we have that $\frac{g_\infty(b)}{\log(g_\infty(b))} = \frac{a}{\log a}$. For $x \geq 4$, the function $\frac{x}{\log(x)}$ is one-to-one. By assumption, $a \geq 4$ and we can conclude $g_\infty(b) = a$ as desired. □

3.2.2 Estimating running time of BKZ in practice

We now review various estimates for the running time of BKZ that exist in the literature. Perhaps the most prominent and widely used estimate is due to Lindner and Peikert [150] who estimate that the runtime (in seconds) of BKZ is

$$\log t_{BKZ}(\delta_0) = \frac{1.8}{\log \delta_0} - 110\,.$$

To convert the estimate to a more general metric we convert to clock cycles. Lindner and Peikert obtained their estimate based on experiments using the implementation of BKZ in the NTL library [204]. The experiments were performed on a 2.3 GHz computer, so the estimate corresponds to a runtime of approximately $2^{\frac{1.8}{\log \delta_0} - 78.9}$ clock cycles.

We argue the Lindner and Peikert model is inaccurate. One reason for this is due to the implementation of the SVP oracle in NTL. This is implemented as enumeration without improvements such as extreme pruning, local block preprocessing, and early termination, which means the SVP oracle requires $2^{\mathcal{O}(k^2)}$ time. According to the Lindner and Peikert model the log of the running time is linear in $\frac{1}{\log \delta_0}$ whereas using Corollary 1 it is clear that the log of the running time is nonlinear in $\frac{1}{\log \delta_0}$. Even deriving an analogous result to Corollary 1 using the less tight bound

$$k \geq g_1\left(\frac{1}{2\log \delta_0}\right) = \frac{1}{2\log \delta_0}$$

we would obtain that the log of the running time is $\Omega\left(\frac{1}{4\log^2 \delta_0}\right)$, which is still nonlinear in $\frac{1}{\log \delta_0}$. A second reason, which we will discuss in Section 4.6, is that applying this model to predict the behaviour of BKZ leads to a subexponential algorithm for solving LWE.

Albrecht *et al.* [9] use data points of Liu and Nguyen [152] to extrapolate a model similar to Lindner and Peikert's [150]. We refer to the model of Albrecht *et al.* [9]

as the delta-squared model. The running time in seconds on a 2.3 GHz computer of BKZ 2.0 is estimated to be

$$\log t_{BKZ}(\delta_0) = \frac{0.009}{\log^2 \delta_0} - 27\,,$$

corresponding to a runtime of $2^{\frac{0.009}{\log^2 \delta_0}+4.1}$ clock cycles. We note that this model also assumes that enumeration in BKZ 2.0 has a complexity of $2^{\mathcal{O}(k^2)}$, and that the running times on which this model is based were not independently verified, which limits their utility.

Chen and Nguyen [65, 64] provide a simulation algorithm for BKZ 2.0 for arbitrarily high block size k, under the assumption that each block behaves as a random basis. They note that this assumption may not hold for block sizes $k < 50$. The algorithm takes as input the logs of the norms of the Gram-Schmidt vectors belonging to the input matrix and a block size k. It outputs the expected logs of the norms of the Gram-Schmidt vectors of the BKZ$_k$-reduced basis as well as the number of rounds ρ needed. The simulation algorithm allows us to calculate the block size k that will be required to obtain the given approximate δ_0 by BKZ 2.0 (see [65, Table 2]). Chen and Nguyen assume the SVP oracle is implemented using a pruned enumeration and they estimate the upper bound of the cost of this, for various values of k, in terms of the number of nodes of the enumeration tree (see [65, Table 3]). Each round of BKZ is estimated to cost $n-k$ enumeration calls, so the total cost of BKZ is estimated to be the number of rounds multiplied by $n-k$ multiplied by the cost of an enumeration call.

In [211] van de Pol and Smart consider the problem of estimating the cost of BKZ from the perspective of using BKZ to solve an LWE instance or some other computational problem in lattices. They assume we have a desired level of security 2^λ; that is, a maximum number of operations an adversary can perform, and a given lattice dimension m. These values are used to find the lowest δ_0 that can be achieved in 2^λ operations, minimising over possible choices of the block size k and the number of rounds $\rho = \rho(k, m, \lambda)$. This is in contrast to an approach where the parameters of the system correspond to a δ_0, which then implies a certain security level. They use data of Chen and Nguyen [65, Table 3] to estimate the cost of one enumeration for a given k and to calculate the total number of enumerations that can be performed for this k, to reach the maximum of 2^λ operations. Note that this means they do not

consider block sizes $k > 250$ as Chen and Nguyen do not give estimates for those. They remark that δ_0 seems to converge to a value depending only on k, corroborating other results in the literature. They note further that the convergence is slower in higher dimension. The approach of van de Pol and Smart was later refined by Lepoint and Naehrig [149].

Alkim *et al.* [19] and Bos *et al.* [39] also consider the cost of BKZ in the setting of estimating security of their respective key exchange schemes. Suppose an attacker uses BKZ with appropriate block size to obtain an output basis of quality such that they expect to be able to break the scheme. It can be argued that this attacker can be expected to make at most a polynomial number of SVP oracle calls, for example by using an early termination strategy [124]. In particular, experiments of Chen [64] suggest that a linear number of calls to the SVP oracle may be sufficient. Alkim *et al.* [19] and Bos *et al.* [39] take a more conservative view and instead imagine that the attacker is successful after just one call to the SVP oracle. In [39] it is argued that this accounts both for potential improvements to BKZ, as well as for a case in which the attacker requires to run BKZ several times and may be able to amortise its cost.

Albrecht [7] estimates the cost of BKZ in clock cycles as

$$t_{BKZ} = 8 \cdot n \cdot \left(2^{0.292k+12.31}\right) .$$

This follows Chen [64] in estimating a linear number of calls to the SVP oracle, namely $c \cdot n$ calls for a constant c, and assumes the SVP oracle is implemented via sieving. The constant $c = 8$ is chosen based on experiments using fplll [88]. The $0.292k$ term is based on the asymptotic estimate of the cost of sieving given by Becker *et al.* [33] and the 12.31 term is based on experiments of Laarhoven [142].

3.3 Estimating lattice reduction in this thesis

Alongside the article [16], Albrecht developed a module in Sage for determining the bit security of LWE instances [6], which we call the *LWE estimator*. The LWE estimator takes as input an LWE instance parameterised by n, α and q and returns estimates for the bit operations required to solve that instance via various algorithms,

SVP oracle	Data source	Log of cost t_k of SVP oracle
Enumeration	[66]	$0.270\,k\log(k) - 1.019\,k + 16.10$
Classical sieve small	[33]	$0.387\,k + 16.4$
Classical sieve asymptotic	[33]	$0.292k + 16.4$
Quantum sieve	[141]	$0.265\,k + 16.4$

Table 3.1: Estimates used in the LWE estimator for the log of the cost in clock cycles $\log(t_k)$ to solve SVP in dimension k.

as well as the amount of memory and number of LWE samples that would be required when using each algorithm. As we will discuss in detail in Chapter 4, lattice reduction is required in many approaches for solving LWE. In this section we discuss how the cost of lattice reduction is estimated in the LWE estimator. We focus on commit f13c4a7, which is the version we will use in the remainder of the thesis.

Let t_k be an estimate for the cost in clock cycles for an SVP oracle in dimension k. In earlier versions of the LWE estimator, for example at the time of publication of [16], the cost of BKZ in dimension n with block size k was estimated in clock cycles as $n \cdot \rho \cdot t_k$. The number of rounds was estimated as $\rho = \frac{n^2}{k^2}\log n$, following [124], giving the estimate in clock cycles as

$$t_{BKZ} = \frac{n^3}{k^2}\log n \cdot t_k\,.$$

In commit f13c4a7 of the LWE estimator the cost of BKZ in dimension n with block size k is estimated in clock cycles as

$$t_{BKZ} = 8 \cdot n \cdot t_k\,,$$

following Albrecht [7]. It is assumed that the SVP oracle is implemented as sieving or enumeration and estimates for t_k that are used are summarised in Table 3.1. The classical sieving estimates are based on the work of Becker *et al.* [33] and the quantum sieving estimate is based on the thesis of Laarhoven [141]. The enumeration estimate is based on the work of Chen and Nguyen [66].

The enumeration estimate, given in the first row of Table 3.1, was derived by fitting $a\,k\log(k) + b\,k + c$ to data in [66, Table 4] and assuming one enumeration costs 200 clock cycles as in [66]. Although BKZ 2.0 does perform local block preprocessing with BKZ-k' before calling the SVP oracle, it is not clear when implemented as in [66]

that it achieves a complexity of $k^{\mathcal{O}(k)}$. This is because [66] does not give sufficient details to decide if preprocessing was performed with $k' = \mathcal{O}(k)$. However, from a cryptanalytical perspective, it is safer to assume it achieves this bound, which is why the curve was fit to $a\,k\log(k) + b\,k + c$ rather than $a\,k^2 + b\,k + c$.

The estimates for classical sieving are given in the second and third rows of Table 3.1. Becker *et al.* [33] report a heuristic asymptotic time complexity of $2^{0.292\,k+o(k)}$ and experimental time complexities of $2^{0.405k-16}$ seconds and $2^{0.387k-15}$ seconds on a 2.83 GHz CPU. The faster experimental estimate is used for small k. The estimate $2^{0.292k+c}$ is used for $k > 90$; that is, the $o(k)$ term is treated as a constant c, which is chosen to be the same as the constant term for small k.

The estimate for quantum sieving is given in the final row of Table 3.1. An asymptotic complexity of quantum sieving of $2^{0.265\,k+o(k)}$ is reported by Laarhoven [141]. Again, the $o(k)$ term is treated as a constant c which is set to the same value as for the other sieving estimates.

While it could be argued that it is unreasonable to ignore the contribution of the $o(k)$ terms in the sieving estimates, we note that Becker *et al.* [33] also fit experimental data to a curve of the form 2^{ak+b} for constants a and b. Furthermore, we note that estimating the cost of sieving in practice has been stressed as an open problem by, for example, Laarhoven and de Weger [143].

Chapter 4

Algorithms for solving LWE

Contents

This chapter discusses approaches for solving the Learning with Errors problem. In particular, we identify three strategies for solving LWE and describe the algorithms available in the literature for solving LWE via these strategies. This chapter is based on material presented in [16].

4.1 Motivation

The hardness of the Learning with Errors problem is considered only asymptotically in much of the literature. Such asymptotic expressions can be useful: for example, they give a good overview of the general behaviour of algorithms for solving LWE. This is illustrated by Herold *et al.* [128], who show that for a polynomial number of

samples, all known algorithms for solving LWE in dimension n have running time at least $2^{\mathcal{O}(n)}$. In addition, asymptotic results can indicate weaker parameter settings. For example, the algorithm of Arora and Ge [25] runs in time $2^{\tilde{\mathcal{O}}(n^{2\xi})}$ for some ξ such that $\alpha q = n^{\xi}$ and so we can conclude that it is subexponential when $\xi < \frac{1}{2}$. This result coincides nicely with the reduction from GapSVP to LWE due to Regev [195], which requires $\alpha q > \sqrt{n}$. As another example, Kirchner and Fouque [139] give an algorithm that is subexponential in the case of a uniform binary error. This indicates that LWE with binary error is somewhat weaker than general LWE, an idea which we will discuss in more detail in the following chapter.

However, asymptotic expressions for the runtime of algorithms may hide logarithmic and constant factors in the exponent. Therefore, using such expressions can make determining secure parameters challenging when designing cryptosystems. When choosing parameters for an LWE-based cryptosystem we must ensure that the underlying LWE instance is hard with respect to a particular security parameter λ. To choose optimal parameters for security we must be able to identify the fastest known way of solving LWE with that choice of parameters and be assured that this attack requires at least 2^{λ} operations. Our goal in this chapter is therefore to determine the concrete hardness of LWE.

In the following section we identify three strategies for solving LWE: firstly, via solving the Short Integer Solutions problem, known as the dual attack; secondly, via solving the Bounded Distance Decoding problem, known as the primal attack; and finally, recovering the secret directly by other means. In the remainder of the chapter, for each of these strategies, we discuss the algorithms that can be used to solve LWE. We consider all possible approaches for solving LWE, including algorithms that may require a very large number of samples m. Indeed, in this chapter we assume the attacker has access to an unlimited number of samples. We therefore parameterise an LWE instance by n, α and q. This is in line with the implementation of the LWE estimator [6] at the time of publication of [16].

For some approaches for solving LWE, it turns out that the optimal attack requires a specific number of samples m, and the attacker having access to more should discard any excess samples. In particular, during the attack the aim will be to find a vector $\mathbf{v}$ with a target norm in a lattice L with a fixed volume $\mathrm{vol}\,(L)$ but a variable

dimension m. In particular, the optimal attack will require m to be chosen such that the following is minimised:

$$\|\mathbf{v}\|_2 = \delta_0^m \operatorname{vol}(L)^{1/m} .$$

In many applications, it holds that $\operatorname{vol}(L) = q^n$ and in this case, this minimal value is given by

$$m = \sqrt{\frac{n \log q}{\log \delta_0}}$$

and is referred to as the *optimal subdimension* [168].

A criticism of allowing any number of samples is that this does not always represent the situation in practice, since there are scenarios in which an attacker attempting to solve LWE would only have access to a limited number of samples. Hence it can be argued [18] that allowing unlimited samples, and thus assuming the attacker is more powerful, leads to overly conservative security estimates. To address this issue, Bindel *et al.* [36] have implemented an extension to the LWE estimator [6] that enables the user to specify a limit on the number of samples m, in addition to the usual parameters n, α and q, and then obtain an estimate of the hardness of the LWE instance in this case.

4.2 Strategies for solving LWE

In this section we discuss three strategies for solving LWE given samples $(\mathbf{A}, \mathbf{c})$. All of the algorithms that we will discuss in the remainder of this chapter follow one of these strategies.

In Section 4.2.1, we discuss the strategy of solving Decision-LWE by finding a short vector $\mathbf{v}$ such that $\mathbf{v}\mathbf{A} = 0$. This strategy solves the Short Integer Solutions problem and is also known as the *dual attack*.

In Section 4.2.2, we discuss the strategy of solving Search-LWE by finding a short $\mathbf{e}$ such that $\mathbf{A}\mathbf{x} = \mathbf{c} - \mathbf{e}$ for some unknown $\mathbf{x}$. This strategy solves the Bounded Distance Decoding problem and is also known as the *primal attack*.

In Section 4.2.3, we discuss the strategy of solving Search-LWE directly by finding an $\mathbf{s}'$ such that $\mathbf{A}\mathbf{s}'$ is close to $\mathbf{c}$.

4.2.1 Solving LWE via solving the Short Integer Solutions problem

The strategy of solving LWE via solving the Short Integer Solutions problem solves Decision-LWE. The goal is to distinguish the case where m samples $(\mathbf{A}, \mathbf{c})$ follow $L_{\mathbf{s},\chi}$, and hence satisfy $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e}$ with $\mathbf{e}_{(i)} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\alpha q}$, from the case where both $\mathbf{A}$ and $\mathbf{c}$ are uniformly random. To do so, we aim to find a short vector $\mathbf{v}$ in the lattice

$$L = \{\mathbf{w} \in \mathbb{Z}_q^m \mid \mathbf{w}\mathbf{A} \equiv 0 \mod q\},$$

which is the dual lattice, scaled by q, of the lattice generated by $\mathbf{A}$. Finding such a $\mathbf{v}$ is exactly solving the Short Integer Solutions problem. To see that such a $\mathbf{v}$ will enable us to distinguish, consider the inner product $\langle \mathbf{v}, \mathbf{c} \rangle$. If $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e}$ then $\langle \mathbf{v}, \mathbf{c} \rangle = \langle \mathbf{v}, \mathbf{e} \rangle$ which approximately follows a Gaussian distribution over $\mathbb{Z}$ considered modulo q, provided not too many of the components of $\mathbf{v}$ are too large. In particular, $\langle \mathbf{v}, \mathbf{e} \rangle$ is often small as both $\mathbf{v}$ and $\mathbf{e}$ are small. On the other hand, if $\mathbf{c}$ is uniform then $\langle \mathbf{v}, \mathbf{c} \rangle$ is uniform over $\mathbb{Z}_q$. So we can indeed distinguish these two cases, thus solving Decision-LWE. We must however ensure $\|\mathbf{v}\|_2$ is suitably short. If $\|\mathbf{v}\|_2$ is too large then the Gaussian distribution of $\langle \mathbf{v}, \mathbf{e} \rangle$ will be too flat to distinguish from random. In particular, we have the following lemma specifying the distinguishing advantage for a given vector $\mathbf{v}$.

Lemma 10 ([150])**.** *Given an LWE instance parameterised by n, α, q and a vector $\mathbf{v}$ of length $\|\mathbf{v}\|_2$ in the scaled dual lattice $L = \{\mathbf{w} \in \mathbb{Z}_q^m \mid \mathbf{w}\mathbf{A} \equiv 0 \mod q\}$, the advantage of distinguishing $\langle \mathbf{v}, \mathbf{e} \rangle$ from random is close to $\exp(-\pi(\|\mathbf{v}\|_2 \cdot \alpha)^2)$.*

Corollary 2. *To obtain an advantage ϵ in solving an LWE instance that is parameterised by n, α and q via the SIS strategy, we require a vector $\mathbf{v}$ of norm $\|\mathbf{v}\|_2 = \frac{1}{\alpha} \cdot \sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}$.*

Stehlé [208] states that a suitably short choice to distinguish $L_{\mathbf{s},\chi}$ from random is $\|\mathbf{v}\|_2 \cdot \alpha q \leq q$; that is, $\|\mathbf{v}\|_2 \leq \frac{1}{\alpha}$. By Lemma 10, choosing $\|\mathbf{v}\|_2 = \frac{1}{\alpha}$ results in an advantage of about $\frac{1}{23}$ to distinguish correctly.

Depending on the algorithm used to obtain the short vector $\mathbf{v}$, it may be advantageous to accept a longer vector as output. This decreases the distinguishing advantage ϵ, but, by the Chernoff bound [72], running the algorithm about $\frac{1}{\epsilon^2}$ times will achieve a success probability close to 1. This may be faster than the alternative, which uses fewer vectors at a higher success probability, but takes significantly longer to obtain these shorter vectors.

4.2.2 Solving LWE via solving Bounded Distance Decoding

The strategy of solving LWE via solving the Bounded Distance Decoding problem (BDD) solves Search-LWE. Given m samples $(\mathbf{A}, \mathbf{c})$ following $L_{\mathbf{s},\chi}$, we may observe that $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e}$ is close to a linear combination of the columns of $\mathbf{A}$. Furthermore, since the noise $\mathbf{e}$ is Gaussian, almost all of the noise is bounded in norm by a small multiple of the standard deviation. Considering the lattice spanned by the columns of $\mathbf{A}$, we can see that $\mathbf{c}$ is a point within a bounded distance from the lattice point $\mathbf{w} = \mathbf{A}\mathbf{s}$. Hence, we may view the LWE instance as an instance of the BDD problem for this lattice. In this case, our solution to the BDD problem would be the lattice point $\mathbf{w}$, from which we may then use linear algebra to recover $\mathbf{s}$ and therefore solve Search-LWE. In the case that $\mathbf{A}$ is not invertible, we call for more samples until it is.

Let ϵ' be the overall target success probability. Depending on the algorithm used, it may be advantageous to run the algorithm independently t times, each with a lower success probability ϵ. Since we are solving a search problem, we will fail overall only if we fail on all of the t attempts. Hence we have $1 - \epsilon' = (1 - \epsilon)^t$ and so an overall success probability of ϵ' is achieved after repeating $t = \frac{\log(1-\epsilon')}{\log(1-\epsilon)}$ times.

4.2.3 Solving LWE via recovering the secret directly

The final strategy is to solve Search-LWE directly by searching for a suitable $\mathbf{s}'$ such that $\|\mathbf{A}\mathbf{s}' - \mathbf{c}\|_2$ is small. While this and the BDD strategy are related by simple linear algebra, since knowing $\mathbf{e}$ trivially allows us to recover $\mathbf{s}$ and vice versa, they differ in which of $\mathbf{e}$ or $\mathbf{s}$ they target.

4.3 Exhaustive search for LWE

The most naive approach for solving LWE is an exhaustive search. This directly solves for $\mathbf{s}$ as in Section 4.2.3.

Theorem 2. *Let an LWE instance be parameterised by n, α and q. The time complexity of solving* Search-LWE *with success probability ϵ with exhaustive search is*

$$m \cdot (2t\alpha q + 1)^n \cdot 2n = 2^{n \log (2t\alpha q+1)+\log n+1+\log m}$$

operations in $\mathbb{Z}_q$. The memory complexity is n and the number of samples required is $n + m$ with

$$m = \frac{\log(1-\epsilon) - n\ \log(2t\alpha q + 1)}{\log(2t\alpha)}$$

for some small parameter $t = \omega(\sqrt{\log n})$.

Proof: Consider $\{-t\alpha q, \ldots, t\alpha q\}$ for $t = \omega(\sqrt{\log n})$. By Lemma 3, an LWE sample has error that falls in this range with high probability. Applying Lemma 4 we obtain an LWE instance with $\mathbf{s}_{(i)} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\alpha q}$; that is, the secret $\mathbf{s}$ follows the same distribution as the error. Therefore we are able to estimate the size of each component $\mathbf{s}_{(i)}$ of the secret $\mathbf{s}$ as $|\mathbf{s}_{(i)}| \leq t\alpha q$ and so to check all possible secrets we must enumerate approximately $(2t\alpha q + 1)^n$ vectors. For each guess $\mathbf{s}'$ and for all available samples $(\mathbf{a}_i, c_i)$ we calculate $x_i = c_i - \langle \mathbf{a}_i, \mathbf{s}' \rangle$ and see if x_i follows the error distribution, which would be the case for the correct secret. For each guess $\mathbf{s}'$ and for each sample we perform about $2n$ operations in $\mathbb{Z}_q$ when computing the inner product, which gives the stated time complexity.

We require n of the overall set of samples to apply Lemma 4. We determine the number of additional samples m required by bounding the probability of a false positive. We know that the correct secret $\mathbf{s}' = \mathbf{s}$ will produce $e_i = c_i - \langle \mathbf{a}_i, \mathbf{s} \rangle$ with $e_i \in \{-t\alpha q, \ldots, t\alpha q\}$ with overwhelming probability. Wrong guesses $\mathbf{s}'$ will produce random elements in $\mathbb{Z}_q$ that land within the acceptable range with probability at most $\frac{\lceil 2t\alpha q \rceil + 1}{q} \approx 2t\alpha$. For an incorrect guess $\mathbf{s}' \neq \mathbf{s}$ to pass the test it must pass for all m samples, which happens with probability $(2t\alpha)^m$. There are $(2t\alpha q + 1)^n - 1$ wrong choices for $\mathbf{s}$. By the union bound, we will accept a false positive with probability $p_f \leq (2t\alpha)^m \cdot (2t\alpha q + 1)^n$. For small failure probability $p_f = 1 - \epsilon$ we obtain the stated bound on the number of additional samples m. □

In the LWE estimator [6] commit f13c4a7, the parameter t is set to $t = \left\lceil 2\sqrt{\ln n} \right\rceil$. Using Lemma 3, the probability of the error falling outside the range $\{-t\alpha q, \ldots, t\alpha q\}$ is at most

$$\frac{2e^{-4\pi \ln n}}{\left(2\sqrt{2\pi}\sqrt{\ln n}\right) \cdot \sqrt{2\pi}} = \frac{2e^{\ln\left(n^{-4\pi}\right)}}{4\pi\sqrt{\ln n}} = \frac{1}{n^{4\pi} \cdot 2\pi\sqrt{\ln n}} .$$

Corollary 3. *Let an LWE instance be parameterised by n, α and q where $q = n^c$ for some constant c and $\alpha q = \sqrt{n}$. Then the time complexity of solving* Search-LWE *with success probability ϵ with exhaustive search is*

$$2^{n \log\left(2t\sqrt{n}+1\right) + \log n + 1 + \log m} .$$

The memory complexity is n. The number of samples required is $m + n$ with

$$m = \frac{\log\left(1-\epsilon\right) - n\log\left(2t\sqrt{n}+1\right)}{\left(\frac{1}{2} - c\right)\log n + \log\left(2t\right)} .$$

4.3.1 Meet-in-the-Middle for LWE

There is a Meet-in-the-Middle (MITM) algorithm to solve LWE, which directly solves for $\mathbf{s}$ as in Section 4.2.3. This is a time-memory trade-off and is therefore a faster method than a naive brute force at the cost of an increased memory requirement. The existence of an MITM algorithm for LWE was mentioned by Bai and Galbraith [29] but a detailed description was not provided.

Theorem 3. *Let an LWE instance be parameterised by n, α and q. If there are $n+m$ samples satisfying $\left(\frac{2t\alpha q+1}{q}\right)^m \cdot \left((2t\alpha q+1)^{\frac{n}{2}} - 1\right) = \mathrm{poly}(n)$ and $3t\alpha m \leq \frac{1}{C}$ for some small parameter $t = \omega(\sqrt{\log n})$, then there is a Meet-in-the-Middle algorithm that solves* Search-LWE *with non-negligible probability that runs in time*

$$(2t\alpha q+1)^{\frac{n}{2}} \cdot \left(m \cdot n + \log\left(\frac{n}{2}\right) + \log\left(\log t\alpha q\right) + \frac{n}{2} \cdot \log\left(2t\alpha q+1\right)\right)$$

and requires memory $\frac{n}{2} \cdot \log\left(t\alpha q\right) \cdot (2t\alpha q+1)^{\frac{n}{2}}$.

Proof: We apply Lemma 4, which costs n samples, to obtain an LWE instance with $\mathbf{s}_{(j)} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z},\alpha q}$. That is, the secret follows the same distribution as the error, and by Lemma 3 has elements in the range $\{-t\alpha q, \ldots, t\alpha q\}$ for $t = \omega(\sqrt{\log n})$ with high probability.

With each of the remaining m samples $(\mathbf{a}_k, \langle \mathbf{a}_k, \mathbf{s} \rangle + e_k)$, we split $\mathbf{a}_k = {\mathbf{a}_k}^l \,||\, {\mathbf{a}_k}^r$ in half. For every possible first half of the secret ${\mathbf{s}_i}^l$, and for each of the m samples $\mathbf{a}_k$, we compute $\left\langle {\mathbf{a}_k}^l, {\mathbf{s}_i}^l \right\rangle$. Let the output of a guess ${\mathbf{s}_i}^l \in \mathbb{Z}_q^{n/2}$ be the vector

$$\mathbf{u}_{{\mathbf{s}_i}^l} = \left(\left\langle {\mathbf{a}_0}^l, {\mathbf{s}_i}^l \right\rangle, \ldots, \left\langle {\mathbf{a}_{m-1}}^l, {\mathbf{s}_i}^l \right\rangle \right) .$$

We store a table T whose entries map $\mathbf{u}_{{\mathbf{s}_i}^l}$ to ${\mathbf{s}_i}^l$. In particular, we index ${\mathbf{s}_i}^l$ by the $(\log q - \log(t\alpha q))$ most significant bits of $\mathbf{u}_{{\mathbf{s}_i}^l}$ in T.

Since the secret follows the error distribution, we expect $(2t\alpha q + 1)^{\frac{n}{2}}$ candidate secret halves ${\mathbf{s}_i}^l$ and for each candidate we must calculate m inner products. Therefore, generating the table costs $m \cdot n \cdot (2t\alpha q + 1)^{\frac{n}{2}}$ operations. We need to store $(2t\alpha q + 1)^{\frac{n}{2}}$ candidates ${\mathbf{s}_i}^l$, each of which has $\frac{n}{2}$ components, each of which requires $\log(t\alpha q)$ space. Hence the memory requirement is $|T| = \frac{n}{2} \cdot \log(t\alpha q) \cdot (2t\alpha q + 1)^{\frac{n}{2}}$.

Once the table is generated and sorted, for each candidate ${\mathbf{s}_j}^r \in \mathbb{Z}_q^{n/2}$ for the second half of the secret, and for each of the samples $\mathbf{a}_k$, we compute $c_k - \langle {\mathbf{a}_k}^r, {\mathbf{s}_j}^r \rangle$. Let the output for a guess ${\mathbf{s}_j}^r$ be the vector

$$\mathbf{v}_{{\mathbf{s}_j}^r} = \left(c_0 - \langle {\mathbf{a}_0}^r, {\mathbf{s}_j}^r \rangle, \ldots, c_{m-1} - \langle {\mathbf{a}_{m-1}}^r, {\mathbf{s}_j}^r \rangle \right) .$$

We take the $(\log q - \log(t\alpha q))$ most significant bits of $\mathbf{v}_{{\mathbf{s}_j}^r}$ and query T. If T returns a value ${\mathbf{s}_i}^l$, we treat ${\mathbf{s}_i}^l \,||\, {\mathbf{s}_j}^r$ as a candidate secret. By construction, if T is queried for the guess ${\mathbf{s}_j}^r$ and returns ${\mathbf{s}_i}^l$, then $\|\mathbf{v}_{{\mathbf{s}_j}^r} - \mathbf{u}_{{\mathbf{s}_i}^l}\| \leq t\alpha q$ since $\mathbf{v}_{{\mathbf{s}_j}^r}$ and $\mathbf{u}_{{\mathbf{s}_i}^l}$ agree except for the least $\log(t\alpha q)$ significant bits. If this process returns no candidate secret, we call for more samples and repeat.

For each guess ${\mathbf{s}_j}^r$, we can query $\mathbf{v}_{{\mathbf{s}_j}^r}$ in

$$\begin{aligned} \log(|T|) &= \log\left(\frac{n}{2} \cdot \log(t\alpha q) \cdot (2t\alpha q + 1)^{\frac{n}{2}} \right) \\ &= \log\left(\frac{n}{2} \right) + \log(\log t\alpha q) + \frac{n}{2} \log(2t\alpha q + 1) \end{aligned}$$

operations by binary search. Since there are $(2t\alpha q + 1)^{\frac{n}{2}}$ possible guesses ${\mathbf{s}_j}^r$, the overall cost of querying is $(2t\alpha q + 1)^{\frac{n}{2}} \cdot \left(\log\left(\frac{n}{2}\right) + \log(\log t\alpha q) + \frac{n}{2} \cdot \log(2t\alpha q + 1) \right)$.

Let the k^{th} component of $\mathbf{v}_{\mathbf{s}^r}$ be $v_{\mathbf{s}^r,k}$ and let the k^{th} component of $\mathbf{u}_{\mathbf{s}^l}$ be $u_{\mathbf{s}^l,k}$. For the correct secret $\mathbf{s} = \mathbf{s}^l \,||\, \mathbf{s}^r$, if the $(\log q - \log(t\alpha q))$ most significant bits of $\mathbf{v}_{\mathbf{s}^r}$ and $\mathbf{u}_{\mathbf{s}^l}$ agree, then

$$v_{\mathbf{s}^r,k} - u_{\mathbf{s}^l,k} = c_k - \langle \mathbf{a}_k^r, \mathbf{s}^r \rangle - \left\langle \mathbf{a}_k^l, \mathbf{s}^l \right\rangle = c_k - \langle \mathbf{a}_k, \mathbf{s} \rangle = e_k \mod q$$

for all k. By Lemma 3 the errors e_k are in the range $\{-t\alpha q, \ldots, t\alpha q\}$ with overwhelming probability. Therefore, if this process returns any candidate secrets ${\mathbf{s}_i}^l \,||\, {\mathbf{s}_j}^r$ at all, then with overwhelming probability, one of them will be the correct secret $\mathbf{s}$.

We need to ensure the $b = (\log q - \log(t\alpha q))$ most significant bits of $\mathbf{v}_{\mathbf{s}^r}$ and $\mathbf{u}_{\mathbf{s}^l}$ agree. We denote this requirement as $MSB_b(\mathbf{v}_{\mathbf{s}^r}) = MSB_b\left(\mathbf{u}_{\mathbf{s}^l}\right)$. For all k, by definition of the LWE samples,

$$\begin{aligned} c_k &= \langle \mathbf{a}_k, \mathbf{s} \rangle + e_k \mod q \\ c_k &= \left\langle {\mathbf{a}_k}^l, \mathbf{s}^l \right\rangle + \langle {\mathbf{a}_k}^r, \mathbf{s}^r \rangle + e_k \mod q \\ c_k - \langle {\mathbf{a}_k}^r, \mathbf{s}^r \rangle &= \left\langle {\mathbf{a}_k}^l, \mathbf{s}^l \right\rangle + e_k \mod q \\ v_{\mathbf{s}^r,k} &= u_{\mathbf{s}^l,k} + e_k \mod q \\ MSB_b(v_{\mathbf{s}^r,k}) &= MSB_b\left(u_{\mathbf{s}^l,k} + e_k\right) . \end{aligned}$$

Hence, it is sufficient to show that $MSB_b\left(u_{\mathbf{s}^l,k} + e_k\right) = MSB_b\left(u_{\mathbf{s}^l,k}\right)$ for all k. Since with overwhelming probability $\|e_k\| \leq t\alpha q$, this holds whenever the inner product $\left\langle {\mathbf{a}_k}^l, \mathbf{s}^l \right\rangle$ takes a value in the range $\{t\alpha q, \ldots, q - 1 - 2t\alpha q\}$. In this range, adding e_k will not cause a wrap around modulo q or influence the b most significant bits. We would like the probability that the inner product is not in this range to be small. Since $\mathbf{a}$ is uniformly random, so is the inner product $\left\langle {\mathbf{a}_k}^l, \mathbf{s}^l \right\rangle$. Therefore the probability that $\left\langle {\mathbf{a}_k}^l, \mathbf{s}^l \right\rangle$ is not in the range $\{t\alpha q, \ldots, q-1-2t\alpha q\}$ is $\frac{t\alpha q + 2t\alpha q}{q} = 3t\alpha$. By the union bound, the probability that this occurs for any of the m samples is less than or equal to $3t\alpha m$. We therefore require that m satisfies $3t\alpha m \leq \frac{1}{C}$ for some constant C.

Consider now the probability of a false positive, that is, a wrong candidate secret ${\mathbf{s}_i}^l$ being suggested for some candidate ${\mathbf{s}_j}^r$. Since $\mathbf{a}_k$ is uniformly random, for any ${\mathbf{s}_i}^l$, we have that $\mathbf{u}_{{\mathbf{s}_i}^l}$ is essentially a random vector where each component takes one of q values. The probability of a wrong candidate ${\mathbf{s}_j}^r$ producing a $\mathbf{v}_{{\mathbf{s}_j}^r}$ matching to a given $\mathbf{u}_{{\mathbf{s}_i}^l}$ is the probability of getting to within $\pm t\alpha q$ on every component. Therefore the probability of a false positive is $\left(\frac{2t\alpha q+1}{q}\right)^m$. There are $(2t\alpha q + 1)^{\frac{n}{2}} - 1$ wrong choices for ${\mathbf{s}_i}^l$. We hence expect to test $\left(\frac{2t\alpha q+1}{q}\right)^m \cdot \left((2t\alpha q + 1)^{\frac{n}{2}} - 1\right)$ candidates per ${\mathbf{s}_j}^r$ and thus we also require that m satisfies

$$\left(\frac{2t\alpha q + 1}{q}\right)^m \cdot \left((2t\alpha q + 1)^{\frac{n}{2}} - 1\right) = \mathrm{poly}(n) .$$

□

In the LWE estimator [6] commit f13c4a7, the required number of samples m is estimated heuristically as follows. The probability of a false positive is approximated as $(2t\alpha)^m$. The number of wrong choices for $\mathbf{s}_i{}^l$ is approximated as $(\alpha q)^{n/2}$. The choice $2n$ is fixed for poly(n). Hence the required number of samples is estimated as

$$m = \frac{\log\left(\frac{2n}{(\alpha q)^{n/2}}\right)}{\log\left(2t\alpha\right)}.$$

To heuristically ensure the second condition on the number of samples, it is then checked that

$$2m\alpha > 1 - \frac{1}{2n}.$$

4.4 The Arora-Ge algorithm

Arora and Ge [25] proposed an algorithm that solves Search-LWE via recovering the secret directly as described in Section 4.2.3. The algorithm works by setting up a system of noise-free nonlinear polynomials of which the secret $\mathbf{s}$ is a root. The system is then solved by linearisation to recover $\mathbf{s}$.

In more detail, the algorithm proceeds by assuming that the error always falls in the range $[-t, t]$ for some $t \in \mathbb{Z}$ such that $d = 2t + 1 < q$. This is a reasonable assumption in the case of a Gaussian error distribution since by Lemma 3 the chance of falling outside this interval drops exponentially fast. Polynomials are constructed from the observation that the error, when falling in this range, is always a root of the polynomial $P(x) = x\prod_{i=1}^{t}(x+i)(x-i)$. Then, we know the secret $\mathbf{s}$ is a root of $P(\mathbf{a}\cdot\mathbf{x} - c)$ constructed from LWE samples. This system of nonlinear equations is solved by linearisation; that is, by replacing each monomial with a new variable and solving the resulting linear system. This approach requires a large number, $\mathcal{O}\left(n^d\right)$, of samples and as we increase the number of samples, we increase the probability that the error falls outside of the interval $[-t, t]$. We then have to increase the range, which in turn requires even more samples. Balancing these two requirements of keeping t low and acquiring enough samples, the overall complexity is given by the following result.

Theorem 4 ([10])**.** *Let an LWE instance be parameterised by n, q and $\sigma = \alpha q$, let ω denote the linear algebra constant, and define $D_{\mathrm{AG}} = 8\,\sigma^2 \log n + 1$. If $D_{\mathrm{AG}} \in o(n)$*

then the Arora-Ge algorithm solves Search-LWE *in time complexity*

$$\mathcal{O}\left(2^{\omega \cdot D_{\mathrm{AG}} \log \frac{n}{D_{\mathrm{AG}}}} \cdot \sigma\, q \log q\right) = \mathcal{O}\left(2^{8\,\omega\,\sigma^2 \log n(\log n - \log(8\,\sigma^2 \log n))}\right)$$

and memory complexity

$$\mathcal{O}\left(2^{2 \cdot D_{\mathrm{AG}} \log \frac{n}{D_{\mathrm{AG}}}} \cdot \sigma\, q \log q\right) = \mathcal{O}\left(2^{16\,\sigma^2 \log n(\log n - \log(8\,\sigma^2 \log n))}\right).$$

If $n \in o(D_{\mathrm{AG}})$ *then the Arora-Ge algorithm solves* Search-LWE *in time complexity*

$$\mathcal{O}\left(2^{\omega\, n \log \frac{D_{\mathrm{AG}}}{n}} \cdot \sigma\, q \log q\right) = \mathcal{O}\left(2^{\omega\, n \log(8\,\sigma^2 \log n) - \omega n \log n}\right)$$

and memory complexity

$$\mathcal{O}\left(2^{2n \log \frac{D_{\mathrm{AG}}}{n}} \cdot \sigma\, q \log q\right) = \mathcal{O}\left(2^{2n \log(8\,\sigma^2 \log n) - 2n \log n}\right).$$

Corollary 4 ([10])**.** *Let an LWE instance be parameterised by* n*,* q *and* $\sigma = \sqrt{n}$*. Then the time complexity of the Arora-Ge algorithm is* $\mathcal{O}\left(2^{(1+\epsilon)\omega n \log\log n}\right)$ *where* $\epsilon = \frac{3}{\log\log n}$*.*

Albrecht *et al.* [10] show that the Arora-Ge algorithm can be improved by using Gröbner basis techniques. As mentioned above, to solve via linearisation as in [25] we require $\mathcal{O}\left(n^d\right)$ samples, but Gröbner basis algorithms will work when fewer samples than this are available at the cost of a more expensive solving step. This approach requires the assumption that random systems behave like semi-regular sequences, a justification for which is given in [10]. In particular, in the case that $\alpha q = \sqrt{n}$, we obtain Theorem 5.

Theorem 5 ([10])**.** *Let* $(\mathbf{a}_i, c_i)$ *for* $i \geq 1$ *be elements of* $\mathbb{Z}_q^n \times \mathbb{Z}_q$ *sampled according to* $L_{\mathbf{s},\chi}$ *with* $\alpha q = \sqrt{n}$ *and let* ω *denote the linear algebra constant. Then there is a heuristic algorithm recovering the secret with time complexity* $\mathcal{O}\left(2^{2.35\omega n + 1.13n}\right)$*, memory complexity* $\mathcal{O}\left(2^{5.85n}\right)$ *and sample complexity* $m = \exp(\frac{\pi}{4} \cdot n)$*.*

4.5 The BKW algorithm

The BKW algorithm, named for its inventors Blum, Kalai and Wasserman [37], was introduced as an algorithm for solving Learning Parity with Noise (LPN). As noted by Regev [195], the BKW algorithm can be adapted to solve LWE.

The BKW algorithm is parameterised by a and $b = \frac{n}{a}$, and solves LWE via the SIS strategy as described in Section 4.2.1. To solve with this strategy, given m samples $(\mathbf{A}, \mathbf{c})$ following $L_{\mathbf{s},\chi}$, we require short vectors $\mathbf{v}_i$ in the scaled dual lattice of the lattice generated by $\mathbf{A}$. The vectors $\mathbf{v}_i$ are obtained by adding elements from a tables where each table is used to find collisions on b components of a row of $\mathbf{A}$.

Albrecht *et al.* [9] gave the first analysis of the BKW algorithm for LWE. In their variant, the vectors $\mathbf{v}_i$ are constructed as follows. Given a sample $\mathbf{a}$; that is, a row of $\mathbf{A}$, the BKW algorithm splits the n components into a blocks each of width b. There are a stages of the algorithm in which tables are created by searching for collisions in the appropriate b coefficients of $\mathbf{a}$. In the first stage, after an appropriate number of samples, we obtain two vectors that agree on $\mathbf{a}_{(0)}, \ldots, \mathbf{a}_{(b-1)}$. The algorithm will then take these and subtract them producing a row with $\mathbf{a}_{(0)} = \cdots = \mathbf{a}_{(b-1)} = 0$ which is stored for use in the next stage. The next stage considers $\mathbf{a}_{(b)}, \ldots, \mathbf{a}_{(2b-1)}$, and so on. This means that the algorithm must maintain a tables of size q^b and its running time is typically dominated by this magnitude.

The $\mathbf{v}_i$ are of length $\sqrt{2^a}$. In the first stage, suppose we find a collision on the first b components. Adding those vectors to clear the first b components in $\mathbf{a}$ produces a $\mathbf{v}_i$ candidate of length $\sqrt{2}$ as we are adding two vectors. Moving on to the next stage, two such vectors are added to clear the next b components, resulting in a $\mathbf{v}_i$ candidate of length $\sqrt{2^2}$, and so on for all a stages.

Albrecht *et al.* [9] give the following complexity for solving Decision-LWE with the BKW algorithm. We note that [9] optimistically gives $m = \frac{\epsilon}{\exp(-\pi\,\alpha^2\,2^a)}$, but by the Chernoff bound we need about $m = \frac{1}{\exp(-\pi\,\alpha^2\,2^a)^2}$ samples to distinguish.

Theorem 6 ([9])**.** *Let $(\mathbf{a}_i, c_i)$ be samples following $L_{\mathbf{s},\chi}$ or a uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$, $0 < b \leq n$ be a parameter and $a = \lceil \frac{n}{b} \rceil$ the addition depth. Then the expected cost (over the randomness of the oracle generating samples) of the BKW algorithm to distinguish $L_{\mathbf{s},\chi}$ from random with success probability ϵ is*

$$\left(\frac{q^b - 1}{2}\right) \cdot \left(\frac{a(a-1)}{2} \cdot (n+1) - \frac{ba(a-1)}{4}\right) - \frac{b}{6}\left(\frac{q^b - 1}{2}\right)\left((a-1)^3 + \frac{3}{2}(a-1)^2 + \frac{1}{2}(a-1)\right)$$

additions/subtractions in $\mathbb{Z}_q$ to produce elimination tables, and

$$\frac{1}{\exp\left(-\pi\,\alpha^2\,2^a\right)^2}\cdot\left(\frac{a}{2}\cdot(n+2)\right)$$

additions/subtractions in $\mathbb{Z}_q$ to produce samples. Furthermore,

$$a\cdot\left\lceil\frac{q^b}{2}\right\rceil+\frac{1}{\exp\left(-\pi\,\alpha^2\,2^a\right)^2}$$

calls to $L_{\mathbf{s},\chi}$ and storage for

$$\left(\frac{q^b}{2}\right)\cdot a\cdot\left(n+1-b\cdot\frac{a-1}{2}\right)$$

elements in $\mathbb{Z}_q$ are needed.

To pick a, which determines a choice for b, recall from Section 4.2.1 that in order to distinguish $L_{\mathbf{s},\chi}$ from random using SIS an appropriately short choice for $\mathbf{v}_i$ is $\|\mathbf{v}_i\|_2\cdot\alpha q=\sqrt{2^a}\cdot\alpha q\leq q$, and hence a suitable choice for a is $a\leq\log(\alpha^{-2})$.

Corollary 5 ([9])**.** *Let $a=-2\log\alpha$ and $b=\frac{n}{a}$. Then the expected cost of the BKW algorithm to distinguish $L_{\mathbf{s},\chi}$ from random is*

$$\begin{aligned}\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)\right)+\mathrm{poly}(n)&\leq q^b\cdot a^2n+\mathrm{poly}(n)\\&=\mathcal{O}\left(q^{n/(-2\log\alpha)}\cdot(-2\log\alpha)^2n\right)\\&=\mathcal{O}\left(2^{n\log q/(-2\log\alpha)}\cdot(-2\log\alpha)^2n\right)\end{aligned}$$

operations in $\mathbb{Z}_q$. Furthermore, $a\cdot\left\lceil\frac{q^b}{2}\right\rceil+\mathrm{poly}(n)$ calls to $L_{\mathbf{s},\chi}$ and storage for $\left(\frac{q^b}{2}\right)\cdot a\cdot n$ elements in $\mathbb{Z}_q$ are needed.

Specialising Corollary 5 with $q=n^c$ and $\alpha q=\sqrt{n}$ we obtain the following time complexity.

Corollary 6 ([9])**.** *Let $q=n^c$, $\alpha q=\sqrt{n}$, $a=-2\log\alpha$ and $b=\frac{n}{a}$. Then the expected cost of the BKW algorithm to distinguish $L_{\mathbf{s},\chi}$ from random is*

$$\begin{aligned}\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)\right)+\mathrm{poly}(n)&\leq q^b\cdot\left(a^2n\right)+\mathrm{poly}(n)\\&=\mathcal{O}\left(2^{n\log q/-2\log\alpha}\cdot\mathrm{poly}(n)\right)\\&=\mathcal{O}\left(2^{\frac{cn\log n}{(2c-1)\log n}}\cdot\mathrm{poly}(n)\right)\\&=2^{\frac{n}{2-(1/c)}}\cdot\mathrm{poly}(n)\end{aligned}$$

operations in $\mathbb{Z}_q$.

Albrecht *et al.* [9] also give a variant of BKW that solves Search-LWE. The idea is that after a number of iterations, each of which clears a subset of the components of the $\mathbf{a}_i$, we are left with a small-dimensional LWE instance with a secret $\mathbf{s}'$ whose components are a subset of the components of the original secret $\mathbf{s}$. Candidates for $\mathbf{s}'$ are tested using the log-likelihood ratio, and once the $\mathbf{s}'$ part of the secret $\mathbf{s}$ is recovered, back substitution is used to recover the next subset of $\mathbf{s}$, and so on. This variant was improved by Duc *et al.* [94] who use a discrete Fourier transform to recover a correct subset $\mathbf{s}'$, arriving at the following complexity result.

Theorem 7 ([94])**.** *Let an LWE instance be parameterised by n, α, and q and let a, $b \in \mathbb{N}$ be such that $a \cdot b = n$. Let C_{FFT} be the small constant in the complexity of the fast Fourier transform computation. Let $0 < \epsilon < 1$ be a targeted success probability and define $\epsilon' = \frac{1-\epsilon}{a}$. For $0 \leq j \leq a-1$ let*

$$m_{j,\epsilon} = 8 \cdot b \cdot \log\left(\frac{q}{\epsilon}\right) \cdot \left(1 - \alpha^2\pi\right)^{-2^{a-j}} .$$

The time complexity to recover the secret $\mathbf{s}$ with probability at least ϵ is $c_1+c_2+c_3+c_4$ where

$$c_1 = \frac{q^b - 1}{2} \cdot \left(\frac{(a-1)(a-2)}{2}(n+1) - \frac{b}{6}\left(a(a-1)(a-2)\right)\right)$$

is the number of additions in $\mathbb{Z}_q$ to produce tables,

$$c_2 = \sum_{j=0}^{a-1} m_{j,\epsilon'} \cdot \frac{a-1-j}{2}(n+2)$$

is the number of additions in $\mathbb{Z}_q$ to recover $\mathbf{s}$,

$$c_3 = 2\left(\sum_{j=0}^{a-1} m_{j,\epsilon'}\right) + C_{FFT} \cdot n \cdot q^b \cdot \log q$$

is the number of operations in $\mathbb{C}$ to prepare and compute the discrete Fourier transforms, and

$$c_4 = (a-1)(a-2) \cdot b \cdot \frac{q^b - 1}{2}$$

is the number of operations in $\mathbb{Z}_q$ for back substitution. Furthermore we require $(a-1)\frac{q^b-1}{2} + m_{0,\epsilon'}$ calls to $L_{\mathbf{s},\chi}$ and storage for

$$\frac{q^b - 1}{2}(a-1)\left(n + 1 - b\frac{a-2}{2}\right) + m_{0,\epsilon}$$

elements in $\mathbb{Z}_q$ and q^b elements in $\mathbb{C}$.

A variant called Coded-BKW was proposed by Guo *et al.* [120], and a similar variant was proposed concurrently by Kirchner and Fouque [139]. Recall that in standard BKW, a vector $\mathbf{a}_1$ is added or subtracted with a second vector $\mathbf{a}_2$ when both $\mathbf{a}_1$ and $\mathbf{a}_2$ agree on a certain range of coordinates, so that many coordinates can be cancelled out at once. This means a smaller LWE instance is produced in a process incurring minimal noise growth. The process is then iterated until a very small dimensional instance is left, which can be easily solved. The main idea in [120, 139] is to find other ways to classify vectors as agreeing, other than having the same values in many adjacent coordinates. As remarked in [139], this is similar to the BKW variant of Albrecht *et al.* [12] known as *lazy modulus switching*, which we will discuss in Section 5.3.

In Coded-BKW, vectors are mapped to the lattice codeword that is nearest in Euclidean norm, and vectors that are mapped to the same codeword are cancelled. This process, known as *quantisation*, introduces noise, which can be controlled so that it is not more than the noise that would have been introduced with a standard BKW procedure. Indeed, the algorithm calls for a certain number of standard BKW operations followed by a number of Coded-BKW operations.

As observed for example by Albrecht [7], in light of these algorithms we can see BKW-type algorithms as a general framework: firstly we create a small dimensional LWE instance via iterations of some quantisation method, then we solve the small dimensional instance via any preferred method.

4.6 Solving Decision-LWE via lattice reduction

Lattice reduction is another means to find short vectors in the scaled dual lattice, enabling us to solve LWE via the SIS strategy as described in Section 4.2.1. We construct this lattice $L = \{\mathbf{w} \in \mathbb{Z}_q^m \mid \mathbf{w}\mathbf{A} \equiv 0 \mod q\}$ from a given $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ as follows. First we compute a basis $\mathbf{B}$ for the left kernel of $\mathbf{A}$ over $\mathbb{Z}_q$, then we extend it by $q\mathbf{I}$. In more detail, with high probability $\mathbf{A}$ is full rank and so $\mathbf{B}$ is an $(m-n) \times m$ matrix. We express $\mathbf{B}$ in row echelon form as $(\mathbf{I}_{m-n} \,|\, \mathbf{B}')$, and hence

obtain a basis for L as

$$\begin{pmatrix} \mathbf{I}_{m-n} & \mathbf{B}' \\ 0 & q\mathbf{I}_n \end{pmatrix}.$$

The lattice L has dimension m, rank m and with high probability volume $\mathrm{vol}(L) = q^n$ [168].

By our convention lattice reduction will return the shortest nonzero vector it found as the first vector $\mathbf{b}_0$ of a reduced basis, which by definition is a short vector in L, so that $\mathbf{b}_0 \mathbf{A} = 0 \mod q$. Heuristically, for a good enough output basis all vectors could be used, as they will all be somewhat short, that is, not too dissimilar in length from each other.

Lemma 11. *Let an LWE instance be parameterised by n, α and q. Any lattice reduction algorithm achieving log root-Hermite factor*

$$\log \delta_0 = \frac{\log^2\left(\frac{1}{\alpha} \cdot \sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}\right)}{4n \log q}$$

leads to an algorithm distinguishing $L_{\mathbf{s},\chi}$ from uniform with advantage ϵ.

Proof: In order to distinguish with advantage ϵ we require $\|\mathbf{v}\|_2 = \frac{1}{\alpha}\sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}$ by Corollary 2. By definition the Hermite factor is $\delta_0^m = \frac{\|\mathbf{v}\|_2}{\mathrm{vol}(L)^{\frac{1}{m}}}$ and with high probability $\mathrm{vol}(L) = q^n$ so using lattice reduction we can achieve $\|\mathbf{v}\|_2 = \delta_0^m \, q^{\frac{n}{m}}$. We choose the optimal subdimension $m = \sqrt{\frac{n \log q}{\log \delta_0}}$ which minimises the quantity $\delta_0^m q^{\frac{n}{m}}$. Rearranging with this value of m, we obtain

$$\log \delta_0 = \frac{\log^2\left(\frac{1}{\alpha} \cdot \sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}\right)}{4n \log q}$$

as our desired log root-Hermite factor. □

Corollary 7. *Let an LWE instance be parameterised by n, α and q where $q = n^c$ for some constant c and $\alpha q = \sqrt{n}$. Any lattice reduction algorithm achieving log root-Hermite factor*

$$\log \delta_0 = \frac{\left(\left(c - \frac{1}{2}\right) \log n + \log \sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}\right)^2}{4cn \log n}$$

can distinguish $L_{\mathbf{s},\chi}$ with advantage ϵ.

Proof: As in the proof of Lemma 11 with high probability we can obtain a vector of norm $\|\mathbf{v}\|_2 = \delta_0^m q^{\frac{n}{m}}$, and we require a vector $\|\mathbf{v}\|_2 = \frac{1}{\alpha}\sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}$ in order to distinguish $L_{\mathbf{s},\chi}$ with advantage ϵ. We therefore require δ_0 such that

$$\begin{aligned}\delta_0^m q^{\frac{n}{m}} &= \frac{1}{\alpha}\sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}} \\ \delta_0^m n^{\frac{cn}{m}} &= n^{c-\frac{1}{2}}\sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}\,.\end{aligned}$$

Again we use the optimal subdimension $m = \sqrt{\frac{n\log q}{\log \delta_0}} = \sqrt{\frac{cn\log n}{\log \delta_0}}$. Rearranging with this value of m, we obtain our required log root-Hermite factor as follows:

$$\begin{aligned}\sqrt{\frac{cn\log n}{\log\delta_0}}\log\delta_0 + \frac{cn}{\sqrt{\frac{cn\log n}{\log\delta_0}}}\log n &= \left(c-\frac{1}{2}\right)\log n + \log\sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}} \\ \frac{2cn\log n}{\left(\left(c-\frac{1}{2}\right)\log n + \log\sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}\right)} &= \sqrt{\frac{cn\log n}{\log\delta_0}} \\ \frac{\left(\left(c-\frac{1}{2}\right)\log n + \log\sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}\right)^2}{4cn\log n} &= \log\delta_0\,.\end{aligned}$$

□

As noted in Section 4.2.1 above, the approach as discussed so far may not be optimal in terms of running time. Given access to sufficiently many samples m it is usually faster to run several lattice reductions, each with a small success probability ϵ, as opposed to running one very strong lattice reduction with target success probability close to 1. Namely, we expect to run lattice reduction with success probability ϵ about $\frac{1}{\epsilon^2}$ times to achieve the overall target success probability $\epsilon' \approx 1$. For example, the LWE estimator commit f13c4a7 with Regev [195] parameters with $n = 128$ gives that the lowest overall running time is obtained using $\epsilon = 0.003906$.

From the discussion above, in order to solve Decision-LWE using lattice reduction, we must either run a very strong lattice reduction once, or run a somewhat strong lattice reduction several times. Albrecht [7] argues that it might be possible to amortise the cost of lattice reduction in the latter case using rerandomisation. That is, it may be sufficient to perform the somewhat strong lattice reduction only once, and we can obtain the other short vectors, which are also required to distinguish

Model	Block size k	log clock cycles
Lattice rule of thumb	$\frac{k}{\log k} = \frac{4n \log q}{\log^2\left(\frac{1}{\alpha}\right)}$	$\mathcal{O}(k)$
Simplified lattice rule of thumb	$\frac{4n \log q}{\log^2\left(\frac{1}{\alpha}\right)}$	$\mathcal{O}\left(\frac{4n \log q}{\log^2\left(\frac{1}{\alpha}\right)}\right)$
Lindner and Peikert [150]	N/A	$\frac{7.2n \log q}{\log^2\left(\frac{1}{\alpha}\right)} - 78.9$
Albrecht *et al.* [9]	N/A	$\frac{0.144n^2 \log^2 q}{\log^4\left(\frac{1}{\alpha}\right)} + 4.1$

Table 4.1: Time complexity for distinguishing $L_{\mathbf{s},\chi}$ from random with advantage $\epsilon \approx \frac{1}{23}$ based on lattice reduction estimates from the literature.

successfully, by rerandomising the output basis. This is done as follows: the basis is rerandomised by multiplication with a sparse unimodular matrix with small entries, and this rerandomised basis is then is given as input to a cheaper lattice reduction, such as LLL. The vector $\mathbf{v}_i$ is then read off as the shortest vector of the basis that is output from the cheaper lattice reduction. This approach loses statistical independence, so requires the heuristic assumption that this issue is negligible. Albrecht [7] provides experimental evidence that this is a reasonable assumption.

4.6.1 Verifying lattice reduction models

Having established the root-Hermite factor δ_0 required to solve an LWE instance via lattice reduction, we can combine it with estimates of the running time of lattice reduction algorithms from the literature to estimate the runtime of solving LWE in this way. In particular, we will show that some lattice reduction estimates in the literature imply a subexponential algorithm for LWE. Algorithms to solve LWE in general are expected to be exponential, so we can conclude that these lattice reduction estimates are inaccurate.

In Tables 4.1 and 4.2 we list estimates for how long it would take lattice reduction algorithms to achieve the target δ_0 for $\sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}} = 1$, that is, success probability $\epsilon \approx \frac{1}{23}$. Considering the right-most column of Table 4.2 it is clear that both the Lindner and Peikert model [150] as well as the simplified lattice rule of thumb would predict a

Model	Block size k	log clock cycles
Lattice rule of thumb	$\frac{k}{\log k} = \frac{4cn\log n}{\left(\left(c-\frac{1}{2}\right)\log n\right)^2}$	$\mathcal{O}(n)$
Simplified lattice rule of thumb	$\frac{4cn\log n}{\left(\left(c-\frac{1}{2}\right)\log n\right)^2}$	$\mathcal{O}\left(\frac{4cn\log n}{\left(\left(c-\frac{1}{2}\right)\log n\right)^2}\right)$
Lindner and Peikert [150]	N/A	$\frac{7.2cn\log n}{\left(\left(c-\frac{1}{2}\right)\log n\right)^2} - 78.9$
Albrecht *et al.* [9]	N/A	$\frac{0.144c^2n^2\log^2 n}{\left(\left(c-\frac{1}{2}\right)\log n\right)^4} + 4.1$

Table 4.2: Time complexity for distinguishing $L_{\mathbf{s},\chi}$ from random with advantage $\epsilon \approx \frac{1}{23}$ in the case that $q = n^c$ and $\alpha = n^{\frac{1}{2}-c}$ for a constant c based on lattice reduction estimates from the literature.

subexponential running time for solving Decision-LWE using lattice reduction.

4.7 Algorithms for solving Bounded Distance Decoding

The approach of solving an LWE instance by viewing it as an instance of the Bounded Distance Decoding (BDD) problem was first suggested by Lindner and Peikert [150]. We describe this approach using Babai's Nearest Plane algorithm [27], which is the most basic way of solving BDD. Let there be m samples $(\mathbf{A}, \mathbf{c})$ of an LWE instance parameterised by n, α and q. We perform lattice reduction on the lattice $L(\mathbf{A}^T)$ to obtain a new basis $\mathbf{B}$ for this lattice, where the quality of this basis is characterised as usual by the root-Hermite factor δ_0. The basis $\mathbf{B}$ and the target vector $\mathbf{c}$ are input to Babai's Nearest Plane algorithm, and assuming the basis is of sufficient quality, the output is the LWE error vector $\mathbf{e}$. One can then subtract $\mathbf{e}$ from $\mathbf{c}$ and use linear algebra to recover the secret $\mathbf{s}$.

Using Lemma 2, the probability that this approach finds the vector $\mathbf{s}$ is given by the probability that the error vector $\mathbf{e}$ lies in the parallelepiped $\mathcal{P}(\mathbf{B}^*)$. So, it can be seen that the success probability is determined by the quality of the lattice reduction. In particular, solving LWE via decoding using Babai's Nearest Plane recovers the

vector $\mathbf{s}$ with probability

$$\prod_{i=0}^{m-1} \operatorname{erf}\left(\frac{\|\mathbf{b}_i^*\|\sqrt{\pi}}{2\alpha q}\right)$$

under the assumption that sampling from a discrete Gaussian is approximately the same as sampling from a continuous Gaussian [150].

Lindner and Peikert [150] suggest an alteration of Babai's algorithm, designed to widen the fundamental parallelepiped in the direction of $\mathbf{b}_i^*$ by a factor of some integer $d_i > 0$, thereby increasing the chance of $\mathbf{e}$ falling inside it. This will find multiple solutions, which can be searched through exhaustively to find the correct solution. This modifies the success probability to

$$\prod_{i=0}^{m-1} \operatorname{erf}\left(\frac{d_i \cdot \|\mathbf{b}_i^*\|\sqrt{\pi}}{2\alpha q}\right) .$$

Given m, n, α and q as above, let d_i be chosen such that the success probability is at least ϵ. Let t_{node} be the number of clock cycles it takes to visit one node and let

$$t_{NP}(\delta_0, \epsilon) = t_{node} \cdot \prod_{i=0}^{m-1} d_i \, .$$

Then the time required for a decoding approach to achieve success probability ϵ can be determined as

$$t_{dec}(\epsilon) = \min_{\delta_0} \left\{ t_{BKZ}(\delta_0) + t_{NP}(\delta_0, \epsilon) \right\} .$$

As noted in Section 4.2.2 we may also opt to repeat the attack several times, each with a lower advantage.

There is no obvious way to analytically determine the optimal d_i to achieve a desired success probability. Lindner and Peikert [150] suggest that the d_i should be chosen to maximise $\min_{1 \le i \le m} (d_i \cdot \|\mathbf{b}_i^*\|)$. On the one hand, with a more reduced basis, the values of d_i can be smaller, so the Nearest Planes algorithm requires less time. On the other hand, the lattice reduction takes significantly more time to achieve a smaller δ_0. We note that in [150, Figure 4] it appears as though $t_{dec}(\epsilon)$ has not been optimised. Lindner and Peikert [150] find values for δ_0 for which the time of a decoding approach is less than an equivalent distinguishing approach, but these values are not necessarily optimal; that is, the lattice reduction step and the decoding step are not always balanced.

Liu and Nguyen [152] note that the algorithm of Lindner and Peikert, as well as Babai's Nearest Plane, can be viewed as a form of pruned enumeration, but with a different rule to the pruned enumeration described by Gama *et al.* [105]. Let $\mathbf{v}$ be a node, $\mathbf{t}$ be a target vector and $\zeta_i(\mathbf{x}) = \frac{\langle \mathbf{x}, \mathbf{b}_i^* \rangle}{||\mathbf{b}_i^*||}$. The pruning of Gama *et al.* keeps nodes with bounded projections whereas the Lindner and Peikert algorithm keeps nodes with bounded coordinates, in particular $|\zeta_i(\mathbf{v} - \mathbf{t})| \leq \frac{d_i ||\mathbf{b}_i^*||}{2}$. Liu and Nguyen note that this can be generalised to arbitrary bounds on coordinates; that is they suggest $|\zeta_i(\mathbf{v} - \mathbf{t})| \leq R_i$ for some parameters R_i not necessarily dependent on the norms of the Gram-Schmidt vectors $\mathbf{b}_i^*$. Liu and Nguyen implement a variant of the Lindner and Peikert algorithm in the context of pruning algorithms, using arbitrary R_i. They also randomise the input basis, allowing them to repeat the algorithm multiple times, which has the result of increasing both the runtime and success probability linearly. Since we assume access to as many samples as required, we do not rely on rerandomisation when estimating complexity. These two factors result in more flexibility in tuning the parameters, and improved results for solving BDD.

However, instead of using the enumeration framework as simply a method to improve the algorithm of Lindner and Peikert [150], Liu and Nguyen [152] go on to directly apply pruned enumeration to solve BDD. This follows the earlier work of Gama *et al.* [105], and uses linear pruning with the bounds $R_k = \sqrt{\frac{k}{m}} R_m$. Over the same parameters used by Lindner and Peikert, this linear pruning is shown to improve on both the original Nearest Plane algorithm and the Lindner and Peikert variant.

Aono *et al.* [21] give an alternative method for pruning, which they refer to as *band pruning*. A node at depth k is pruned if the projected length is outside the range $[L_k, R_k]$, where the bounds L_k and R_k can be efficiently computed given a desired success probability. Herold *et al.* [128] propose a generalised framework for enumeration techniques in the decoding approach, and show that the approaches of [150, 202, 105] achieve the same running time asymptotically.

The LWE estimator [6] provides an estimate of the cost of the Lindner and Peikert decoding attack [150], and does not provide an estimate of other variants, such as [152, 21]. The Lindner and Peikert estimate that $t_{node} = 2^{15.1}$ clock cycles [150] is used. For comparison, Gama *et al.* [105] achieve $0.94 \cdot 10^7 \approx 2^{23}$ enumerations per

second on a 1.86 GHz machine, which corresponds to $t_{node} = 2^{7.9}$ clock cycles. Since there is no closed formula to determine the optimal d_i, in the LWE estimator the values d_i are increased one by one. The lattice reduction and enumeration steps are balanced.

4.8 Solving LWE via unique Shortest Vector Problem

Another approach for solving an instance of BDD is by reducing it to an instance of the γ-unique Shortest Vector Problem (γ-uSVP). Albrecht *et al.* [13] were the first to consider the complexity of solving LWE in this way.

To reduce BDD to γ-uSVP the embedding technique of Kannan [136] is used. Recall that the usual lattice of consideration when solving LWE via the BDD strategy is

$$L(\mathbf{A}) = \{\mathbf{A}\,\mathbf{u} \mid \mathbf{u} \in \mathbb{Z}_q^n\}\,,$$

the lattice generated by the columns of the matrix $\mathbf{A}$ in the LWE instance $(\mathbf{A}, \mathbf{c})$. The idea is to embed $L(\mathbf{A})$ into a higher-dimensional lattice $L(\mathbf{B})$ with γ-uSVP structure. In particular, $\mathbf{B}$ is constructed as

$$\mathbf{B} = \begin{pmatrix} \tilde{\mathbf{A}} & \mathbf{c} \\ 0 & t \end{pmatrix},$$

where $t = \mathrm{dist}(\mathbf{c}, L(\mathbf{A}))$ is the *embedding factor* and $\tilde{\mathbf{A}}$ is a basis for the q-ary lattice spanned by the columns of $\mathbf{A}$. Lyubashevsky and Micciancio [156] show that if the embedding factor $t < \frac{\lambda_1(L(\mathbf{A})))}{2\gamma}$ then L($\mathbf{B}$) contains a γ-unique shortest vector, $\mathbf{c}' = (\mathbf{e}, -t)$. If we can find this vector, we can take the first m components to recover $\mathbf{e}$, hence solving the BDD instance.

To solve a γ-uSVP instance, we may reduce the problem to κ-Hermite Shortest Vector Problem (κ-HSVP). Intuitively, a lattice with uSVP structure has one direction in which its shortest vector is somewhat shorter than all other directions. A sufficiently good lattice reduction algorithm, for example, can produce a vector so short it must be in this special direction. More precisely, Lovász [154] showed that any algorithm that can solve κ-HSVP, such as a lattice reduction algorithm, can be used linearly many times to solve approximate SVP with approximation factor κ^2. A solution

to κ^2-approximate SVP would be a vector $\mathbf{v}$ such that $\|\mathbf{v}\|_2 \leq \kappa^2 \lambda_1(L)$. For a lattice with κ^2-uSVP structure, any vector $\mathbf{w}$ that is not a shortest vector and that is independent of the shortest vectors satisfies $\|\mathbf{w}\|_2 \geq \lambda_2(L) > \kappa^2 \lambda_1(L)$. So, we must have $\mathbf{v}$ is a multiple of a shortest vector, and hence we have solved γ-uSVP for $\gamma = \kappa^2$. Ling *et al.* [151] improve on this result and show that, for N the dimension of the lattice, if $\kappa > \sqrt{N}$ then any algorithm solving κ-HSVP can be used to solve γ-uSVP for $\gamma \approx \sqrt{N}\kappa$.

The above discussion concerns theoretical results, but we are concerned with using lattice reduction in practice to solve LWE via uSVP. In particular, there are two estimates in the literature for the quality of lattice reduction needed to solve LWE in this way in practice. The more recent estimate is due to Alkim *et al.* [19], and the older estimate is due to Gama and Nguyen [104]. In this thesis we use the latter estimate, following [13, 16], as implemented in commit f13c4a7 of the LWE estimator [6].

Gama and Nguyen [104] observe that lattice reduction of a quality given by a δ_0 such that $\frac{\lambda_2(L)}{\lambda_1(L)} > \tau \delta_0^m$ is sufficient in order to solve a uSVP instance with some probability depending on the value τ, which is taken to be a constant. For a success probability of 0.1, experimental results in [13, 104] show that $\tau \leq 0.4$.

Albrecht *et al.* [13] give the following lemma, under the assumption that $t = \|\mathbf{e}\|_2$.

Lemma 12 (Lemma 2 in [13])**.** *Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, let $\alpha q > 0$ and let $\epsilon' > 1$. Let $\mathbf{e} \in \mathbb{Z}_q^m$ such that each component is drawn from χ and considered mod q. Assuming the Gaussian heuristic for $L(\mathbf{A})$, and that the rows of $\mathbf{A}$ are linearly independent over $\mathbb{Z}_q$, we can create an embedding lattice with $\frac{\lambda_2}{\lambda_1}$-gap greater than*

$$\frac{\min\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\}}{\frac{\epsilon' \alpha q \sqrt{m}}{\sqrt{\pi}}}$$

with probability greater than $1 - \left(\epsilon' \cdot \exp\left(\frac{1-(\epsilon')^2}{2}\right)\right)^m$.

Proof: By construction, $\lambda_1(L(\mathbf{B})) = \|(\mathbf{e}, -t)\|_2$ and $\lambda_2(L(\mathbf{B})) = \min\{q, \lambda_1(L(\mathbf{A}))\}$.

We see that

$$\begin{aligned}\lambda_1(L(\mathbf{B})) &= \|(\mathbf{e}, -t)\|_2 \\ &= \|(\mathbf{e}, -\|\mathbf{e}\|_2)\|_2 \\ &= \sqrt{\|\mathbf{e}\|_2^2 + \|\mathbf{e}\|_2^2} \\ &= \sqrt{2}\,\|\mathbf{e}\|_2\,.\end{aligned}$$

By the Gaussian heuristic for $L(\mathbf{A})$, $\lambda_1(L(\mathbf{A})) = \mathrm{vol}\,(L(\mathbf{A}))^{1/m} \cdot \sqrt{\frac{m}{2\pi e}}$. Since the rows of $\mathbf{A}$ are linearly independent, $\mathrm{vol}\,(L(\mathbf{A})) = q^{m-n}$. Hence we can estimate

$$\lambda_2(L(\mathbf{B})) = \min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}.$$

Therefore we can construct an embedding lattice with

$$\frac{\lambda_2(L(\mathbf{B}))}{\lambda_1(L(\mathbf{B}))} = \frac{\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}}{\sqrt{2}\,\|\mathbf{e}\|_2}.$$

Let $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$ be the standard deviation of the error distribution. We can bound the norm of the error vector as follows [155, Lemma 4.4 (3)]: we have that

$$\|\mathbf{e}\|_2 \leq \epsilon' \cdot \sigma \cdot \sqrt{m}$$

with probability greater than $1 - \left(\epsilon' - \exp\left(\frac{1-(\epsilon')^2}{2}\right)\right)^m$. Hence we have a gap

$$\frac{\lambda_2(L(\mathbf{B}))}{\lambda_1(L(\mathbf{B}))} \geq \frac{\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}}{\sqrt{2}\epsilon' \cdot \sigma \cdot \sqrt{m}} = \frac{\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}}{\sqrt{2}\epsilon' \cdot \frac{\alpha q}{\sqrt{2\pi}} \cdot \sqrt{m}} = \frac{\min\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\}}{\frac{\epsilon'\alpha q\sqrt{m}}{\sqrt{\pi}}}$$

with probability $1 - \left(\epsilon' - \exp\left(\frac{1-(\epsilon')^2}{2}\right)\right)^m$. □

Lemma 12 results in an overly pessimistic estimate for the cost of solving LWE via reduction to uSVP for two reasons. Firstly, the size of $\|\mathbf{e}\|_2$ is determined via an upper bound, but we can also estimate its size using the expectation. Secondly, in practice one would set $t = 1$ rather than $t = \|\mathbf{e}\|_2$. We typically have $\tau \approx 0.3$ in this case [13]. Göpfert [116] provides a more refined estimate.

Lemma 13 ([116])**.** *Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, let $\alpha q > 0$ and let $\epsilon' > 1$. Let $\mathbf{e} \in \mathbb{Z}_q^m$ such that each component is drawn from χ and considered mod q. Assuming the Gaussian*

heuristic for $L(\mathbf{A})$, and that the rows of $\mathbf{A}$ are linearly independent over $\mathbb{Z}_q$, we can create an embedding lattice with λ_2/λ_1-gap greater than

$$\frac{\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}}{\sqrt{m}\cdot\frac{\alpha q}{\sqrt{2\pi}}}\,.$$

Proof: By construction, $\lambda_1(L(\mathbf{B})) = \|(\mathbf{e}, -t)\|_2$ and $\lambda_2(L(\mathbf{B})) = \min\{q, \lambda_1(L(\mathbf{A}))\}$. By the argument given for Lemma 12 we can estimate $\lambda_2(L(\mathbf{B})) = \min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}$. We determine $\lambda_1(L(\mathbf{B}))$ as follows, using $t = 1$:

$$\begin{aligned}\lambda_1(L(\mathbf{B})) &= \|(\mathbf{e}, -1)\|_2 \\ &= \sqrt{\|\mathbf{e}\|_2^2 + 1} \\ &\approx \sqrt{\|\mathbf{e}\|_2^2} \\ &= \|\mathbf{e}\|_2\,.\end{aligned}$$

Therefore we can construct an embedding lattice with

$$\frac{\lambda_2(L(\mathbf{B}))}{\lambda_1(L(\mathbf{B}))} = \frac{\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}}{\|\mathbf{e}\|_2}\,.$$

Since $\mathbf{e}$ is a vector of dimension m with components chosen from a Gaussian distribution with standard deviation $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$ we can estimate

$$\|\mathbf{e}\|_2 = \sqrt{m}\cdot\sigma\,.$$

Hence we have a gap of

$$\frac{\lambda_2(L(\mathbf{B}))}{\lambda_1(L(\mathbf{B}))} = \frac{\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}}{\sqrt{m}\cdot\sigma} = \frac{\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}}{\sqrt{m}\cdot\frac{\alpha q}{\sqrt{2\pi}}}\,.$$

□

We note that Lemma 12 and Lemma 13 are the same up to a factor of $\epsilon'\cdot\sqrt{2}$ in the denominator, which comes from the differing estimates of $\|\mathbf{e}\|_2$ and $\lambda_1(L(\mathbf{B}))$. Using Lemma 13 we can now determine the δ_0 required to solve LWE via reduction to uSVP. Recall that we require a gap of $\frac{\lambda_2(L(\mathbf{B}))}{\lambda_1(L(\mathbf{B}))} > \tau\delta_0^m$. We have two cases, depending on the value of $\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}$. In the first case, in which we have $\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\} = q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}$, we can state the following lemma.

Lemma 14. *Let an LWE instance be parameterised by n, α and q. Then any lattice reduction algorithm achieving log root-Hermite factor*

$$\log \delta_0 = \frac{\log^2 (\tau\alpha\sqrt{e})}{4n \log q}$$

can be used to solve LWE via reduction to uSVP for some fixed $\tau \leq 1$.

Proof: Albrecht *et al.* [13] show that for a fixed δ_0 the optimal subdimension is $m = \sqrt{\frac{n \log q}{\log \delta_0}}$. From the above discussion we require a δ_0 determined by the following:

$$\begin{aligned} \frac{\lambda_2(L(\mathbf{B}))}{\lambda_1(L(\mathbf{B}))} &= \tau\delta_0^m \\ \frac{q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}}{\sqrt{m} \cdot \frac{\alpha q}{\sqrt{2\pi}}} &= \tau\delta_0^m \\ -n \log q = \frac{n \log q}{\log \delta_0} \log \delta_0 &+ \sqrt{\frac{n \log q}{\log \delta_0}} \log \left(\tau\alpha\sqrt{e}\right). \end{aligned}$$

Rearranging the above expression gives the required δ_0. □

In [16], an analogous argument using Lemma 12 shows that the required log root-Hermite factor is $\log \delta_0 = \frac{\log^2 \left(\epsilon'\tau\alpha\sqrt{2e}\right)}{4n \log q}$ for some $\epsilon' > 1$. This differs from the estimate given in Lemma 14 by a $\epsilon' \cdot \sqrt{2}$ term in the numerator, which corresponds to the difference between Lemma 12 and Lemma 13.

Corollary 8. *Let an LWE instance be parameterised by n, α and q where $q = n^c$ for some constant c and $\alpha q = \sqrt{n}$. Any lattice reduction algorithm achieving log root-Hermite factor*

$$\log \delta_0 = \frac{\left(\left(c - \frac{1}{2}\right) \log n - \log (\tau\sqrt{e})\right)^2}{4cn \log n}$$

can be used to solve LWE for some fixed $\tau \leq 1$.

In the second case we have $\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\} = q$ and we can state the following lemma.

Lemma 15. *Let an LWE instance be parameterised by n, α and q. Let $W(\cdot)$ denote the Lambert W function. Any lattice reduction algorithm achieving log root-Hermite factor*

$$\log \delta_0 = \frac{1}{4n^2 \ln^2 q} \cdot \left(W\left((-2n \ln q) \cdot \left(\sqrt{n \log q}\right) \cdot \frac{(\tau\alpha)^2}{2\pi}\right)\right)^2$$

can be used to solve LWE via reduction to uSVP for some fixed $\tau \leq 1$.

Proof: From the above discussion we require a δ_0 determined by the following:

$$\begin{aligned}\tau\delta_0^m &= \frac{\lambda_2(L(\mathbf{B}))}{\lambda_1(L(\mathbf{B}))} \\ \tau\delta_0^m &= \frac{q}{\sqrt{m}\cdot\frac{\alpha q}{\sqrt{2\pi}}}\,.\end{aligned}$$

We assume that the optimal subdimension is $m = \sqrt{\frac{n\log q}{\log\delta_0}}$. We therefore require:

$$\begin{aligned}\delta_0^m\sqrt{m} &= \frac{\sqrt{2\pi}}{\tau\alpha} \\ \delta_0^{4m}m^2 &= \frac{4\pi^2}{(\tau\alpha)^4} \\ \frac{4\pi^2}{(\tau\alpha)^4}\log\delta_0 &= \delta_0^{4\sqrt{\frac{n\log q}{\log\delta_0}}}n\log q \\ 4n\sqrt{\log\delta_0}\log q + \log(n\log q) &= \log\left(\frac{4\pi^2}{(\tau\alpha)^4}\right) + \log(\log\delta_0)\,.\end{aligned}$$

Let $x = \log\delta_0$. We now solve this for x:

$$\begin{aligned}4n\sqrt{x}\log q + \log(n\log q) &= \log\left(\frac{4\pi^2}{(\tau\alpha)^4}\right) + \log x \\ \log\left(q^{4n\sqrt{x}}\cdot(n\log q)\right) &= \log\left(x\cdot\frac{4\pi^2}{(\tau\alpha)^4}\right) \\ e^{4n\sqrt{x}\ln q}\cdot(n\log q) &= x\cdot\frac{4\pi^2}{(\tau\alpha)^4} \\ (n\log q)\cdot\frac{(\tau\alpha)^4}{4\pi^2} &= x\cdot e^{-4n\ln q\sqrt{x}}\,.\end{aligned}$$

Setting $x = y^2$, we require:

$$\begin{aligned}(n\log q)\cdot\frac{(\tau\alpha)^4}{4\pi^2} &= y^2\cdot e^{(-4n\ln q)y} \\ \left(\sqrt{n\log q}\right)\cdot\frac{(\tau\alpha)^2}{2\pi} &= y\cdot e^{(-2n\ln q)y} \\ (-2n\ln q)\cdot\left(\sqrt{n\log q}\right)\cdot\frac{(\tau\alpha)^2}{2\pi} &= (-2n\ln q)\cdot y\cdot e^{(-2n\ln q)y}\,.\end{aligned}$$

Let $z = (-2n\ln q)\cdot y$. We solve for z using the Lambert W function [3]:

$$\begin{aligned}z\cdot e^z &= (-2n\ln q)\cdot\left(\sqrt{n\log q}\right)\cdot\frac{(\tau\alpha)^2}{2\pi} \\ z &= W\left((-2n\ln q)\cdot\left(\sqrt{n\log q}\right)\cdot\frac{(\tau\alpha)^2}{2\pi}\right)\end{aligned}$$

We trace back through our substitutions to arrive at a value for $\log \delta_0$:

$$\begin{aligned}
(-2n \ln q) \cdot y &= W\left((-2n \ln q) \cdot \left(\sqrt{n \log q}\right) \cdot \frac{(\tau\alpha)^2}{2\pi}\right) \\
y^2 &= \frac{1}{4n^2 \ln^2 q} \cdot \left(W\left((-2n \ln q) \cdot \left(\sqrt{n \log q}\right) \cdot \frac{(\tau\alpha)^2}{2\pi}\right)\right)^2 \\
x &= \frac{1}{4n^2 \ln^2 q} \cdot \left(W\left((-2n \ln q) \cdot \left(\sqrt{n \log q}\right) \cdot \frac{(\tau\alpha)^2}{2\pi}\right)\right)^2 \\
\log \delta_0 &= \frac{1}{4n^2 \ln^2 q} \cdot \left(W\left((-2n \ln q) \cdot \left(\sqrt{n \log q}\right) \cdot \frac{(\tau\alpha)^2}{2\pi}\right)\right)^2 .
\end{aligned}$$

□

Alkim *et al.* [19] propose an alternative success condition for lattice reduction in solving LWE via reduction to uSVP. For an LWE instance parameterised by n, α, q and m with standard deviation $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$, let $d = m + n + 1$. The requirement is that

$$\sqrt{k} \cdot \sigma \leq \delta_0^{2k-d} \cdot \mathrm{vol}\,(L)^{\frac{1}{d}} .$$

Albrecht *et al.* [14] provide experimental evidence that when used to solve LWE via uSVP, lattice reduction behaves according to the estimate of Alkim *et al.* [19]. In particular this implies that for some parameter settings, BKZ with a smaller blocksize than would be required according to the Gama and Nguyen [104] estimate will be successful. Hence, a less costly lattice reduction would succeed in solving LWE via uSVP than predicted as in [104, 13, 16, 116] and security claims justified by this line of work may be overly optimistic [14]. Since commit e7c9c59, the estimate for the cost of the uSVP approach in the LWE estimator [6] is implemented according to the estimate of Alkim *et al.* [19].

Chapter 5

Algorithms for solving small variants of LWE

Contents

In this chapter we turn our attention to algorithms for solving variants of LWE with unusually small secret or error. We firstly survey algorithms that are applicable in the case of a small secret. We then focus on the variant of LWE with binary error. We show that this variant is vulnerable to the Howgrave-Graham attack on NTRU, known as the hybrid attack, which is a combination of lattice techniques and a Meet-in-the-Middle approach. We describe other approaches to solving LWE in the case of binary error, and compare the performance of the hybrid attack with these other approaches. This chapter is based on material presented in [16, 50, 52].

5.1 Introduction to LWE with small secrets

In this chapter we consider variants of LWE where the secret is unusually small, which we call *small secret* variants. We first make more precise what we mean by a small secret variant of LWE. In the most general definition of LWE, the secret $\mathbf{s}$ is a vector with components $\mathbf{s}_{(i)}$ chosen uniformly at random from $\mathbb{Z}_q$. By Lemma 4, an LWE instance can always be transformed into an instance where the secret components follow the error distribution, so in general we expect to see components of the secret $\mathbf{s}_{(i)}$ of size at most $t\alpha q$ for some $t = \omega(\sqrt{\log n})$. For small secret variants, we are considering a secret with components much smaller than this: for example, bounded in norm by a small constant.

In particular, we characterise a small secret LWE instance by n, α, q, and ψ, where ψ is the distribution of $\mathbf{s}_{(i)}$. In many applications based on LWE [100, 121, 51, 70], the secret is chosen such that the $\mathbf{s}_{(i)}$ have norm at most 1. This choice is typically for efficiency or performance reasons. For example, Buchmann *et al.* [51] are motivated by lightweight applications for the Internet of Things. Fan and Vercauteren [100] suggest the use of a small secret as an optimisation in their Fully Homomorphic Encryption scheme, which improves the noise growth behaviour in the ciphertexts and may improve performance by enabling a higher depth computation to be homomorphically evaluated without having to choose larger parameters.

To improve noise growth behaviour in homomorphic encryption further, in addition to using a small secret, we could use a sparse secret; that is, a secret with low Hamming weight. This is done for example in HElib [121, 122, 123] which implements the BGV scheme [44]. HElib recommends a Hamming weight $h = 64$ [121], while the dimension n of the secret is typically at least 10^3 or 10^4 in applications (see, for example, [109]). Sparse secrets have also been considered outside of homomorphic encryption applications: motivated by practicality, Cheon *et al.* [67] propose a PKE scheme based on a variant of LWE where the secret $\mathbf{s}$ is sparse but its components $\mathbf{s}_{(i)}$ are not necessarily small in norm.

A prominent example is the variant where the components $\mathbf{s}_{(i)}$ are chosen uniformly from $\{0, 1\}$. This variant is termed *binary secret* [45] and its hardness has been considered in some detail. It has been shown [45, 166] that an LWE instance with a

binary secret $\mathbf{s}$ and dimension $n \log q$ is at least as hard as general LWE in dimension n. On the other hand, Bai and Galbraith [29] conclude from experiments that increasing only to dimension $n \log \log n$ may be sufficient.

Small secret variants of LWE can arise as one of two cases; we consider both in the remainder of this chapter. In both cases the secret $\mathbf{s}$ is small, but in the first case the error is of a typical size, whereas in the second case the error is also small.

As an example of the the first case, we could consider instances parameterised by n, α, q, and ψ where $\alpha q = \sqrt{n}$ and ψ is the uniform distribution on $\{-1, 0, 1\}$. In the case of a small or sparse secret, modifications can be made to the generic algorithms for LWE described in Chapter 4. For example, the algorithm of Arora and Ge [25] can be adapted to the small secret case as follows. A small secret is encoded by adding equations of the form

$$\prod_{i=0}^{k-1} (x - j_i)$$

where k is the cardinality of the support for ψ and j_i are the elements of the support.

In Section 5.2 we discuss solving small secret LWE by exhaustive search. In Section 5.3 we discuss variants of the BKW algorithm [37] that apply in the case of a small secret. In Section 5.4 we introduce an algorithm of Bai and Galbraith [29] that is applicable in the case of a small secret. In Section 5.5 we discuss various improvements possible when using lattice reduction techniques to solve LWE in the case of a small or sparse secret, including the work of Albrecht [7]. This part of the chapter is based on material presented in [16].

As an example of the second case, with very small error, we could consider instances where the errors are chosen uniformly from a narrow range. Since the distribution of the secret can be assumed to follow the error distribution, this also leads to a small secret. Following [50, 52] we focus on the variant of LWE with binary error; that is, the error is chosen uniformly at random from $\{0, 1\}$. We give a thorough introduction to LWE with binary error in Section 5.6.

Algorithms for solving general LWE instances can be applied in the binary error case. In addition to algorithms for standard LWE, we may also be able to apply algorithms

for the *Inhomogeneous Short Integer Solution* problem [5, 111]. We refer the reader to Bai *et al.* [30] for a discussion of these algorithms. In Sections 5.7, 5.8 and 5.9 we discuss respectively the Meet-in-the-Middle algorithm for LWE with binary error, distinguishing LWE with binary error from uniform via lattice reduction, and solving LWE with binary error via reduction to uSVP. In Section 5.10 we describe in detail the hybrid attack for LWE with binary error and in Section 5.11 we compare the hybrid attack with other approaches.

5.2 Exhaustive search for small secret LWE

In Section 4.3 we saw that an LWE instance parameterised by n, α and q can be solved by exhaustive search by checking all the vectors within a sphere of radius $t\alpha q$, for some parameter $t = \omega(\sqrt{\log n})$, since $t\alpha q$ is essentially the size of components of the secret. In the case of a small secret instance, parameterised by n, α, q, and ψ, we can ignore the exact distribution of ψ and simply exhaustively search over the support of ψ. For example, the support may be the set $\{-1, 0, 1\}$, and so we check all possible $\mathbf{s}$ with $\mathbf{s}_{(i)}$ chosen from this set. By the same argument as in Theorem 2, if we have m LWE samples with $\mathbf{s}_{(i)} \in \{-1, 0, 1\}$ then exhaustive search will take time $m \cdot 3^n \cdot (2n) = 2^{n \log 3 + \log n + 1 + \log m}$.

Using the same argument as in Theorem 3, whatever cost we would expect for exhaustive search, which depends on the support of ψ, by applying a Meet-in-the-Middle strategy we may achieve essentially the same speed up as we would do in the case of a general LWE instance. For example, if the components $\mathbf{s}_{(i)}$ are selected from $\{-1, 0, 1\}$ then a Meet-in-the-Middle strategy will take time $\mathcal{O}\left(3^{n/2}\right)$ and require $\mathrm{poly}(n) \cdot 3^{n/2}$ memory.

As observed by Bai and Galbraith [29] we can combine exhaustive search with other algorithms to improve the complexity. For example, we may guess r components of the secret, leaving us with a small secret LWE instance of reduced dimension $n - r$. We may then run our favourite algorithm to solve small secret LWE on this lower dimensional instance, which will be faster. Using this strategy any algorithm discussed below can be turned into an algorithm that has at most the cost of exhaustive search.

5.3 Small secret variants of the BKW algorithm

Several variants of the BKW algorithm [37] are applicable in the small secret case, such as [12, 120, 139]. All these variants follow the idea of repeated quantisation; that is, cancelling vectors that are close to each other on some components, then solving a low dimensional (easy) resulting LWE instance. The algorithms [120, 139], which apply to general LWE instances and were discussed in Section 4.5, are particularly effective in the case of small secret. For example, the variant of Kirchner and Fouque [139] solves LWE with binary secret in subexponential time.

The variant of Albrecht *et al.* [12] requires small secret instances because the quantisation is achieved through a variant of modulus switching. This is termed *lazy modulus switching*; that is, only modulus switching when necessary. Lazy modulus switching means, for example, searching for collisions mod p but remaining in $\mathbb{Z}_q$ when doing arithmetic on the rows. An optional preprocessing step termed *unnatural selection* is additionally proposed.

Theorem 8 ([12])**.** *Let an LWE instance be parameterised by n, α, and q and let $b \in \mathbb{Z}$ with $1 \leq b \leq n$. Define $a = \lceil \frac{n}{b} \rceil$ and pick a pair (p, m^*), which parameterise respectively the modulus to be switched to and the unnatural selection. Then BKW with lazy modulus switching and unnatural selection costs*

$$\frac{p^b}{2}\left(\frac{a(a-1)}{2}(n+1)\right) + (m+m^*)na$$

additions in $\mathbb{Z}_q$ and $\frac{ap^b}{2} + m + m^*$ calls to $L_{\mathbf{s},\chi}$.

In particular, if $q = n^c$ for some small $c \geq 1$, $\alpha q = \sqrt{n}$ and $b = \frac{n}{\log n}$, recall that standard BKW has complexity $\mathcal{O}\left(2^{cn} \cdot n \log^2 n\right)$. Let $0 < d \leq 1$ be a constant. It is shown in [12] that the complexity of solving is $\mathcal{O}\left(2^{n\left(c+\frac{\log d}{\log n}\right)} \cdot n \log^2 n\right)$ if naive modulus switching is used, whereas an approach using lazy modulus switching gives a complexity of $\mathcal{O}\left(2^{n\left(c+\frac{\log d - \frac{1}{2}\log\log n}{\log n}\right)} \cdot n \log^2 n\right)$.

5.4 Solving small secret LWE via unique Shortest Vector Problem

Bai and Galbraith [29] show that for a secret with very small components, such as a binary secret, we may embed our LWE lattice into a lattice with uSVP structure, similar to the approach described in Section 4.8 using Kannan embedding. The target short vector is $(\mathbf{s} \,||\, \mathbf{e} \,||\, 1)$, which contains among its components those of $\mathbf{s}$, in contrast to the target short vector $(\mathbf{e} \,||\, 1)$ of the Kannan embedding case. This enables us to take advantage of the smallness of $\mathbf{s}$, which leads to a more efficient approach.

In particular, the smallness of $\mathbf{s}$ as compared with the size of the error is exploited. If $\|\mathbf{s}\|_2 \ll \|\mathbf{e}\|_2$, we may rescale the lattice into which the instance is embedded, increasing its volume. This increases the δ_0 that is sufficient to solve the instance, and hence a less expensive lattice reduction is required. In the case $\mathbf{s}_{(i)} \leftarrow \{-1, 0, 1\}$, after an appropriate rescaling, the volume of the lattice is increased by σ^n, where $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$ is the standard deviation of the error. In the case $\mathbf{s}_{(i)} \leftarrow \{0, 1\}$ the volume is increased by $(2\sigma)^n$. Bai and Galbraith observe that, perhaps surprisingly, modulus switching does not improve their algorithm. This is because modulus switching results in a smaller rescaling factor.

We now state the cost of Bai and Galbraith's approach, assuming that the embedding factor $t = \|\mathbf{e}\|_2$ is used. We note that both Bai and Galbraith [29] and the LWE estimator [6] use $t = 1$, as in the Kannan embedding case.

Lemma 16. *Let a small secret LWE instance be parameterised by n, α and q with $\mathbf{s}_{(i)} \xleftarrow{\$} \{a, \ldots, b\}$. Let $\xi = 2/(b-a)$ and let $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$. Any lattice reduction algorithm achieving log root-Hermite factor*

$$\log \delta_0 = \frac{\left(\log(\frac{q}{\sigma}) - \log(2\tau\sqrt{\pi e})\right)^2 \cdot \log(\frac{q}{\sigma})}{n(2\,\log(\frac{q}{\sigma}) - \log \xi)^2}$$

solves LWE by reducing BDD to uSVP for some fixed $\tau \leq 1$ if we have that

$$(q^m(\xi\sigma)^n)^{\frac{1}{m+n}} \cdot \sqrt{\frac{m+n}{2\pi e}} \leq q$$

where $m = m' - n = \sqrt{\frac{n(\log q - \log \sigma)}{\log \delta_0}} - n$.

Proof: We follow the discussion of Bai and Galbraith [29, Section 6.2]. For a secret sampled from the interval $[a, \ldots, b]$, by scaling and rebalancing we can increase the volume by a factor of $(\xi\sigma)^n$. Assuming that $(q^m(\xi\sigma)^n)^{\frac{1}{m+n}}\sqrt{\frac{m+n}{2\pi e}} \leq q$, we can construct a lattice with a gap $\frac{\lambda_2}{\lambda_1} = \frac{(q^m(\xi\sigma)^n)^{1/(m+n)}\sqrt{\frac{m+n}{2\pi e}}}{\sqrt{2\,(m+n)}\cdot\sigma}$.

Let $m' = m + n$. By the same argument as in Section 4.8 we expect to be able to solve this uSVP instance when $\frac{\lambda_2}{\lambda_1} \geq \tau\delta_0^{m'}$. Therefore we require δ_0 satisfying:

$$\begin{aligned}
\frac{\left(q^{m'-n}\,(\xi\sigma)^n\right)^{\left(\frac{1}{m'}\right)}}{\sqrt{4\pi e}\,\sigma} &= \tau\,\delta_0^{m'} \\
\left(1-\frac{n}{m'}\right)\log q + \frac{n}{m'}\log(\xi\sigma) &= m'\log\delta_0 + \log\sigma + \log\left(\tau\sqrt{4\pi e}\right) \\
\log\delta_0 &= \frac{m'\,\left(\log(\frac{q}{\sigma}) - \log\left(\tau\sqrt{4\pi e}\right)\right) + n\log(\xi) - n\log\left(\frac{q}{\sigma}\right)}{(m')^2}\,.
\end{aligned}$$

By [29, Lemma 1] the optimal value of m' is $m' = \sqrt{\frac{n(\log q - \log\sigma)}{\log\delta_0}}$. We conclude:

$$\begin{aligned}
\log\delta_0 &= \frac{\sqrt{\frac{n(\log q-\log\sigma)}{\log\delta_0}}\,\left(\log(\frac{q}{\sigma}) - \log\left(\tau\sqrt{4\pi e}\right)\right) + n\log(\xi) - n\log\left(\frac{q}{\sigma}\right)}{\frac{n(\log q-\log\sigma)}{\log\delta_0}} \\
\log\delta_0 &= \frac{\left(\log\left(\frac{q}{\sigma}\right) - \log\left(\tau\sqrt{4\pi e}\right)\right)^2\cdot\log(\frac{q}{\sigma})}{n\left(2\log\left(\frac{q}{\sigma}\right) - \log\xi\right)^2}\,.
\end{aligned}$$

□

For example, specialising Lemma 16 to $\mathbf{s}_{(i)} \xleftarrow{\$} \{-1, 0, 1\}$ and hence $\xi = 1$ gives

$$\log\delta_0 = \frac{\left(\log\alpha\tau\sqrt{2e}\right)^2}{4n\left(\log q - \log\frac{\alpha q}{\sqrt{2\pi}}\right)}\,.$$

5.5 Solving Decision-LWE via lattice reduction for small or sparse secret

For an LWE instance parameterised by n, α, and q and with a small secret, we may apply modulus switching and consider the instance mod p where $p < q$. This allows for a larger root-Hermite factor δ_0 than would be required for an instance parameterised by the same n, α, and q and with a secret where $\mathbf{s}_{(i)}$ is chosen at random from

$\mathbb{Z}_q$. After modulus switching, the transformed instance has an error that is slightly larger and its distribution is no longer exactly a discrete Gaussian. Nonetheless, heuristically, algorithms that solve LWE still solve these LWE-like problem instances and so we assume that after modulus switching, we have an LWE instance characterised by n, $\sqrt{2}\alpha$ and p. So, when we have a small secret we may obtain a speed up by modulus switching before performing the lattice reduction required for several approaches for solving LWE. For example, consider the dual attack solving Decision-LWE by lattice reduction as described in Section 4.6. As with a general secret, we assume the size of the small vector that we aim to output is $\|\mathbf{v}\|_2 = \frac{1}{\sqrt{2}\alpha} \cdot \sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}$.

Lemma 17. *Let a small secret LWE instance be characterised by n, α, q and ψ. Let p be such that $\left\|\left\langle \frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s} \right\rangle\right\|_2 \approx \frac{p}{q} \cdot \|e\|_2$. Any lattice reduction achieving log root-Hermite factor*

$$\log \delta_0 = \frac{\left(\log\left(\frac{1}{\sqrt{2}\alpha} \cdot \sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}\right)\right)^2}{4n \log p}$$

can distinguish the small secret LWE instance from uniform with probability ϵ.

Proof: Using Lemma 6, after modulus switching the LWE instance is parametrised by n, $\sqrt{2}\alpha$, p. Applying Lemma 11 with these parameters we obtain the result claimed. □

Corollary 9. *Let a small secret LWE instance be characterised by n, α, q, ψ. Suppose $\alpha q = \sqrt{n}$, $q = n^c$ for some small constant c and ψ is such that the standard deviation of the elements in the secret $\mathbf{s}$ is σ_s. Any lattice reduction achieving log root-Hermite factor*

$$\log \delta_0 = \frac{\left(\log\left(\frac{1}{\sqrt{2}n^{\frac{1}{2}-c}} \cdot \sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}\right)\right)^2}{4n\left(\log\left(\frac{\sqrt{\pi}}{\sqrt{6}}\sigma_s\right) + c\log n\right)}$$

can distinguish the small secret LWE instance from uniform with probability ϵ.

Proof: From Lemma 6 we have $p = \frac{\sigma_s}{\alpha}\sqrt{\frac{2\pi n}{12}} = \frac{\sigma_s\sqrt{2}\sqrt{\pi}\sqrt{n}}{\sqrt{12}\alpha} = \frac{\sqrt{\pi}}{\sqrt{6}}\sigma_s n^c$. Applying Lemma 17 we obtain

$$\log \delta_0 = \frac{\left(\log\left(\frac{1}{\sqrt{2}\alpha} \cdot \sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}\right)\right)^2}{4n \log p} = \frac{\left(\log\left(\frac{1}{\sqrt{2}n^{\frac{1}{2}-c}} \cdot \sqrt{\frac{\ln\left(\frac{1}{\epsilon}\right)}{\pi}}\right)\right)^2}{4n \log\left(\frac{\sqrt{\pi}}{\sqrt{6}}\sigma_s n^c\right)},$$

from which the result follows. □

Albrecht [7] proposes variants of the distinguishing attack in the case of a small secret. The idea is that rather than solving LWE via finding a solution to the SIS problem $\mathbf{v} \cdot \mathbf{A} \equiv 0 \mod q$ as described in Section 4.2, it is sufficient to find solutions to

$$\mathbf{v} \cdot \mathbf{A} \equiv \mathbf{x} \mod q\,,$$

for some small $\mathbf{x} \bmod q$. In particular we want to find short vectors $(\mathbf{w}, \mathbf{y})$ in the lattice

$$L = \{(\mathbf{v}, \mathbf{x}) \quad | \quad \mathbf{v} \cdot \mathbf{A} \equiv \mathbf{x} \mod q\}\,.$$

Given such a $(\mathbf{w}, \mathbf{y})$ and an LWE sample $(\mathbf{A}, \mathbf{c})$ we hope to distinguish

$$\langle \mathbf{w}, \mathbf{c} \rangle = \langle \mathbf{w}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \rangle = \langle \mathbf{y}, \mathbf{s} \rangle + \langle \mathbf{w}, \mathbf{e} \rangle$$

from uniform. It is clear that we require LWE with a sufficiently small secret $\mathbf{s}$ so that $\langle \mathbf{y}, \mathbf{s} \rangle + \langle \mathbf{w}, \mathbf{e} \rangle$ can be distinguished from uniform, since if a component $\mathbf{s}_{(i)}$ could take any value in $\mathbb{Z}_q$ then this would not be possible. In many small secret LWE instances used in practice, we have that $\|\mathbf{s}\|_2$ is much smaller than $\|\mathbf{e}\|_2$. In that case we can rebalance L using the technique of Bai and Galbraith [29], so the two summands are approximately equal in size.

Lemma 18 ([7]). *Let a small secret LWE instance be parameterised by n, α, q, and ψ where ψ is the distribution producing secrets $\mathbf{s}$ with components $\mathbf{s}_{(i)}$ chosen uniformly at random from $\{-1, 0, 1\}$, with the additional guarantee that at most h components are nonzero. Any lattice reduction achieving log root-Hermite factor*

$$\log \delta_0 = \frac{\log\left(\frac{2n \log^2 \epsilon}{\pi \alpha^2 h}\right)}{8n}$$

can distinguish this small secret LWE instance with advantage ϵ.

Albrecht [7] also provides a technique that can take advantage of the sparseness of a secret. Given an LWE instance $(\mathbf{A}, \mathbf{c})$ one can always split $\mathbf{A}$ into $[\mathbf{A}_0 | \mathbf{A}_1]$ where $\mathbf{A}_1$ contains the last n' columns, and try to find a short vector in

$$L = \{\mathbf{v} \in \mathbb{Z}^m \quad | \quad \mathbf{v} \cdot \mathbf{A}_0 \equiv 0 \mod q\}\,.$$

If $n' = 0$ then this is simply the distinguishing attack described in Section 4.6. A short vector $\mathbf{v} \in L$ produces an LWE sample in dimension n' with noise the size of

original noise multiplied by the expectation of the norm of $\mathbf{v}$. If we choose n' very small, the new instance will be easy to solve and the overall cost of this approach will be less than solving the n dimensional instance directly. That is, the cost of solving the instance in dimension n' together with the cost of finding a short vector $\mathbf{v}$ is less than solving the instance in dimension n.

If the secret $\mathbf{s}$ is also sparse, then it may be possible to choose larger n', and we can also accept some probability of failure and repeat if necessary. In particular, we choose a random permutation to transform $\mathbf{A}$ into $[\mathbf{A}_0|\mathbf{A}_1]$ and we hope that all (respectively most of) the secret components corresponding to $\mathbf{A}_1$ happen to be zero, which occurs with some probability. Then we can (respectively with some exhaustive search component) treat $n' > 0$ as if it were $n' = 0$. That is, we only need the cost of lattice reduction in dimension $n - n'$ (respectively followed by at most the cost of the exhaustive search) and we can still succeed in dimension n.

5.6 Introduction to LWE with binary error

In the remainder of this chapter we turn our attention to the variant of LWE where the error is chosen uniformly from $\{0, 1\}$. We term this variant LWE with binary error.

The hardness of LWE with binary error has been considered in some detail and depends on the number of samples m. It is well known (see for example [92, 166, 10]) that LWE with binary error can be solved in polynomial time using the algorithm of Arora and Ge [25] when the number of samples is $m = \Omega\left(n^2\right)$. Furthermore, Albrecht *et al.* [10] show that the problem can be solved in subexponential time if the attacker has access to $m = \Omega\left(n \log\log n\right)$ samples. Kirchner and Fouque [139] show that the problem can be solved in subexponential time using $m = n$ samples. On the other hand, Micciancio and Peikert [166, Theorem 1.2] give a reduction from worst-case lattice problems to LWE with binary error when $m = n(1 + \Omega\left(1/\log(n)\right)$ and $q \geq n^{\mathcal{O}(1)}$.

Decoding techniques (as described in Section 4.7) can be applied to LWE with binary error. By Lemma 2 the error vector can be recovered using Babai's algorithm if and

only if it lies within $P(\mathbf{B}^*)$. It is shown in [50, 52] that the success probability of decoding using Lindner and Peikert's variant [150] of Babai's algorithm can be estimated as

$$\prod_{i=1}^{m}\left(1-\frac{2}{B(\frac{m-1}{2},\frac{1}{2})}\int_{-1}^{\max(-r_i,-1)}(1-t^2)^{\frac{m-3}{2}}dt\right)$$

where $r_i = d_i\frac{\delta_0^{-2(i-1)+m}q^{\frac{m-n}{m}}}{2\sqrt{m/4}}$ and d_i is as discussed in Section 4.7.

The algorithm of Albrecht *et al.* [10], which uses Gröbner basis methods to improve the algorithm of Arora and Ge [25], can be applied to LWE with binary error. In particular, Albrecht *et al.* [10, Theorem 7] show that if there are $m = 2n$ samples there is an algorithm solving LWE with binary error in time $\mathcal{O}(n^2 \cdot 2^{0.43\omega n})$.

In the following sections we discuss other algorithms that can be used to solving LWE with binary error, including the hybrid attack [50, 52]. We then conclude the chapter with a comparison of the hybrid attack with these other approaches.

5.6.1 Modelling LWE with binary error as general LWE

It will sometimes be convenient to model the error distribution in an LWE with binary error instance as if it were a general LWE instance where the errors come from some discrete Gaussian distribution of standard deviation σ and centre μ that very often outputs a 0 or 1 and rarely outputs a different value. From Chapter 4 we can conclude that for most approaches to solve LWE, the complexity depends on the dimension n and αq, which specifies the size of the errors. That is, most of the algorithms only make use of the fact that the errors are small, rather than the fact that they are discrete Gaussian. The fact that the centre of the discrete Gaussian is 0 also does not affect the algorithms. We can therefore argue that it is reasonable to model a binary error in this way as some discrete Gaussian distribution.

Let X be a distribution on $\{0,1\}$, then X is Bernoulli so that $Pr(X=1)=p$ and $Pr(X=0)=1-p$. Its expectation is $\mathrm{E}[X]=p$ and its variance is $\mathrm{Var}[X]=p(1-p)$. We model this as a Gaussian Y that has mean p and standard deviation $p^{\frac{1}{2}}(1-p)^{\frac{1}{2}}$ such that its expectation and variance are the same as for X. If $p=\frac{1}{2}$ then X is the uniform distribution on $\{0,1\}$ and in this case we model this error distribution

as a Gaussian with mean $\frac{1}{2}$ and standard deviation $\sigma = \frac{1}{2}$. Since the standard deviation of a discrete Gaussian LWE error distribution satisfies $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$, we obtain the following heuristic.

Heuristic 3. *An n, q-LWE with binary error instance can be modelled by an n, q, α-LWE instance where $\alpha = \frac{\sqrt{2\pi}}{2q}$.*

5.7 The Meet-in-the-Middle attack for LWE with binary error

We adapt the analysis in Section 4.3 to determine an upper bound on the complexity of a Meet-in-the-Middle attack on LWE with binary error. We obtain the analogous result to Theorem 3.

Theorem 9. *Let an LWE instance with binary error be parameterised by n and q. If there are $m + n$ samples satisfying $\frac{m}{q} < \frac{1}{C}$ for some constant $C > 1$ and*

$$\left(\frac{2}{q}\right)^m \cdot \left(2^{\frac{n}{2}} - 1\right) = \text{poly}(n)\,,$$

then there is Meet-in-the-Middle algorithm that solves Search-LWE *with binary error with non-negligible probability that runs in time*

$$2^{\frac{n}{2}} \cdot \left(mn + \log \frac{n}{2} + \frac{n}{2}\right)$$

and requires memory $n \cdot 2^{\frac{n}{2}-1}$.

Proof: We apply Lemma 4, which costs n samples, to obtain an LWE instance where the secret follows the error distribution, so that the secret has elements in $\{0, 1\}$.

With each of the remaining m samples $(\mathbf{a}_k, \langle \mathbf{a}_k, \mathbf{s} \rangle + e_k)$, we split $\mathbf{a}_k = {\mathbf{a}_k}^l \,||\, {\mathbf{a}_k}^r$ in half. For every possible first half of the secret ${\mathbf{s}_i}^l$, and for each of the m samples $\mathbf{a}_k$, we compute $\left\langle {\mathbf{a}_k}^l, {\mathbf{s}_i}^l \right\rangle$. Let the output of a guess ${\mathbf{s}_i}^l \in \{0,1\}^{n/2}$ be the vector

$$\mathbf{u}_{{\mathbf{s}_i}^l} = \left(\left\langle {\mathbf{a}_0}^l, {\mathbf{s}_i}^l \right\rangle, \ldots, \left\langle {\mathbf{a}_{m-1}}^l, {\mathbf{s}_i}^l \right\rangle\right) .$$

We store a table T whose entries map $\mathbf{u}_{{\mathbf{s}_i}^l}$ to ${\mathbf{s}_i}^l$. In particular, we index ${\mathbf{s}_i}^l$ by the $\log q - 1$ most significant bits of $\mathbf{u}_{{\mathbf{s}_i}^l}$ in T.

Since the secret follows the error distribution, we expect $2^{\frac{n}{2}}$ candidate secret halves ${\mathbf{s}_i}^l$ and for each candidate we must calculate m inner products. Therefore, generating the table costs $m \cdot n \cdot 2^{\frac{n}{2}}$ operations. We need to store $2^{\frac{n}{2}}$ candidates ${\mathbf{s}_i}^l$, each of which has $\frac{n}{2}$ components, each of which requires 1 bit of space. Hence the memory requirement is $|T| = \frac{n}{2} \cdot 1 \cdot 2^{\frac{n}{2}}$.

Once the table is generated and sorted, for each candidate ${\mathbf{s}_j}^r \in \{0,1\}^{n/2}$ for the second half of the secret, and for each of the samples $\mathbf{a}_k$, we compute $c_k - \langle {\mathbf{a}_k}^r, {\mathbf{s}_j}^r \rangle$. Let the output for a guess ${\mathbf{s}_j}^r$ be the vector

$$\mathbf{v}_{{\mathbf{s}_j}^r} = \left(c_0 - \langle {\mathbf{a}_0}^r, {\mathbf{s}_j}^r \rangle, \ldots, c_{m-1} - \langle {\mathbf{a}_{m-1}}^r, {\mathbf{s}_j}^r \rangle\right) .$$

We take the $\log q - 1$ most significant bits of $\mathbf{v}_{{\mathbf{s}_j}^r}$ and query T. If T returns a value ${\mathbf{s}_i}^l$, we treat ${\mathbf{s}_i}^l \,||\, {\mathbf{s}_j}^r$ as a candidate secret. By construction, if T is queried for the guess ${\mathbf{s}_j}^r$ and returns ${\mathbf{s}_i}^l$, then $\|\mathbf{v}_{{\mathbf{s}_j}^r} - \mathbf{u}_{{\mathbf{s}_i}^l}\| \leq 1$ since $\mathbf{v}_{{\mathbf{s}_j}^r}$ and $\mathbf{u}_{{\mathbf{s}_i}^l}$ agree except for the least significant bit. If this process returns no candidate secret, we call for more samples and repeat.

For each guess ${\mathbf{s}_j}^r$, we can query $\mathbf{v}_{{\mathbf{s}_j}^r}$ in

$$\begin{aligned} \log(|T|) &= \log\left(\frac{n}{2} \cdot 2^{\frac{n}{2}}\right) \\ &= \log\frac{n}{2} + \frac{n}{2} \end{aligned}$$

operations by binary search. Since there are $2^{\frac{n}{2}}$ possible guesses ${\mathbf{s}_j}^r$, the overall cost of querying is $2^{\frac{n}{2}} \cdot \left(\log\frac{n}{2} + \frac{n}{2}\right)$.

Let the k^{th} component of $\mathbf{v}_{\mathbf{s}^r}$ be $v_{\mathbf{s}^r,k}$ and let the k^{th} component of $\mathbf{u}_{\mathbf{s}^l}$ be $u_{\mathbf{s}^l,k}$. For the correct secret $\mathbf{s} = \mathbf{s}^l \,||\, \mathbf{s}^r$, if $\log q - 1$ most significant bits of $\mathbf{v}_{\mathbf{s}^r}$ and $\mathbf{u}_{\mathbf{s}^l}$ agree, then

$$v_{\mathbf{s}^r,k} - u_{\mathbf{s}^l,k} = c_k - \langle \mathbf{a}_k^r, \mathbf{s}^r \rangle - \left\langle \mathbf{a}_k^l, \mathbf{s}^l \right\rangle = c_k - \langle \mathbf{a}_k, \mathbf{s} \rangle = e_k \mod q$$

for all k. Therefore, if this process returns any candidate secrets ${\mathbf{s}_i}^l \,||\, {\mathbf{s}_j}^r$ at all, then with overwhelming probability, one of them will be the correct secret $\mathbf{s}$.

We need to ensure the $b = \log q - 1$ most significant bits of $\mathbf{v}_{\mathbf{s}^r}$ and $\mathbf{u}_{\mathbf{s}^l}$ agree. We denote this requirement as $MSB_b\left(\mathbf{v}_{\mathbf{s}^r}\right) = MSB_b\left(\mathbf{u}_{\mathbf{s}^l}\right)$. For all k, by definition of

the LWE samples,

$$\begin{aligned} c_k &= \langle \mathbf{a}_k, \mathbf{s} \rangle + e_k \mod q \\ c_k &= \left\langle {\mathbf{a}_k}^l, \mathbf{s}^l \right\rangle + \langle {\mathbf{a}_k}^r, \mathbf{s}^r \rangle + e_k \mod q \\ c_k - \langle {\mathbf{a}_k}^r, \mathbf{s}^r \rangle &= \left\langle {\mathbf{a}_k}^l, \mathbf{s}^l \right\rangle + e_k \mod q \\ v_{\mathbf{s}^r,k} &= u_{\mathbf{s}^l,k} + e_k \mod q \\ MSB_b\left(v_{\mathbf{s}^r,k}\right) &= MSB_b\left(u_{\mathbf{s}^l,k} + e_k\right) . \end{aligned}$$

Hence, it is sufficient to show that $MSB_b\left(u_{\mathbf{s}^l,k} + e_k\right) = MSB_b\left(u_{\mathbf{s}^l,k}\right)$ for all k. Since $e_k \in \{0,1\}$, this holds whenever the inner product $\left\langle {\mathbf{a}_k}^l, \mathbf{s}^l \right\rangle$ takes a value in the range $\{0,\ldots,q-2\}$. In this range, adding e_k will not cause a wrap around modulo q. We would like the probability that the inner product is not in this range to be small. Since $\mathbf{a}$ is uniformly random, so is the inner product $\left\langle {\mathbf{a}_k}^l, \mathbf{s}^l \right\rangle$. Therefore the probability that $\left\langle {\mathbf{a}_k}^l, \mathbf{s}^l \right\rangle$ is not in the range $\{0,\ldots,q-2\}$ is $\frac{1}{q}$. By the union bound, the probability that this occurs for any of the m samples is less than or equal to $\frac{m}{q}$. We require m such that $\frac{m}{q} \leq \frac{1}{C}$ for some constant C.

Consider now the probability of a false positive, that is, a wrong candidate secret ${\mathbf{s}_i}^l$ being suggested for some candidate ${\mathbf{s}_j}^r$. Since $\mathbf{a}_k$ is uniformly random, for any ${\mathbf{s}_i}^l$, we have that $\mathbf{u}_{{\mathbf{s}_i}^l}$ is essentially a random vector where each component takes one of q values. The probability of a wrong candidate ${\mathbf{s}_j}^r$ producing a $\mathbf{v}_{{\mathbf{s}_j}^r}$ matching to a given $\mathbf{u}_{{\mathbf{s}_i}^l}$ is the probability of differing by at most 1 on every component. Therefore the probability of a false positive is $\left(\frac{2}{q}\right)^m$. There are $2^{\frac{n}{2}}-1$ wrong choices for ${\mathbf{s}_i}^l$. We hence expect to test $\left(\frac{2}{q}\right)^m \cdot \left(2^{\frac{n}{2}}-1\right)$ candidates per ${\mathbf{s}_j}^r$ and thus have the additional requirement on m that

$$\left(\frac{2}{q}\right)^m \cdot \left(2^{\frac{n}{2}}-1\right) = \mathrm{poly}(n) .$$

□

5.8 Solving Decision-LWE with binary error via lattice reduction

We can solve Decision-LWE with binary error via lattice reduction as in Section 4.6. For this approach we must find a short vector $\mathbf{v}$ in the scaled dual lattice

$$L = \{\mathbf{w} \in \mathbb{Z}_q^m \mid \mathbf{w}\mathbf{A} \equiv 0 \mod q\}.$$

We now determine how short such a vector $\mathbf{v}$ must be.

Lemma 19. *Let an LWE instance with binary error be parameterised by n and q. Let $\mathbf{v} \in L$ be a vector of length*

$$\|\mathbf{v}\|_2 = q \cdot \frac{\sqrt{2\ln\left(\frac{1}{\epsilon}\right)}}{\pi}.$$

Then $\mathbf{v}$ can distinguish the LWE instance from random with advantage ϵ.

Proof: Using Heuristic 3 we can model the error distribution as a Gaussian with mean $\mu = \frac{1}{2}$ and standard deviation $\sigma = \frac{1}{2}$. For the standard deviation of an LWE error distribution we have $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$ and so $\alpha^2 = \frac{2\pi\sigma^2}{q^2} = \frac{\pi}{2q^2}$. By Lemma 10 we can distinguish an LWE instance parameterised by n, α and q with advantage $\epsilon \approx \exp(-\pi(\|\mathbf{v}\|_2 \cdot \alpha)^2)$. Assuming equality, we obtain the following requirement on $\mathbf{v}$ to distinguish with advantage ϵ:

$$\begin{aligned} -\pi(\|\mathbf{v}\|_2 \cdot \alpha)^2 &= \ln \epsilon \\ \|\mathbf{v}\|_2 &= \frac{q\sqrt{2\ln\left(\frac{1}{\epsilon}\right)}}{\pi}. \end{aligned}$$

□

Lemma 20. *Let an LWE instance with binary error be parameterised by n and q. Any lattice reduction algorithm achieving log root-Hermite factor*

$$\log \delta_0 = \frac{\left(\log(q) + \log\left(\frac{\sqrt{2\ln\left(\frac{1}{\epsilon}\right)}}{\pi}\right)\right)^2}{4n\log(q)}$$

can distinguish the LWE with binary error instance with advantage ϵ.

Proof: With high probability the lattice L has rank m and volume q^n [168]. By definition of the Hermite factor we therefore have $\|\mathbf{v}\|_2 = \delta_0^m q^{\frac{n}{m}}$. By Lemma 19 we require $\|\mathbf{v}\|_2 = q \cdot \frac{\sqrt{2\ln\left(\frac{1}{\epsilon}\right)}}{\pi}$ to distinguish with advantage ϵ. We assume that the number of samples m available is large enough to use the optimal subdimension $m = \sqrt{\frac{n \log q}{\log \delta_0}}$. In this case the root-Hermite factor we require to distinguish with advantage ϵ can be determined as follows:

$$\begin{aligned}
\delta_0^m q^{\frac{n}{m}} &= q \cdot \frac{\sqrt{2\ln\left(\frac{1}{\epsilon}\right)}}{\pi} \\
m^2 \log \delta_0 + n \log q &= m\left(\log q + \log\left(\frac{\sqrt{2\ln\left(\frac{1}{\epsilon}\right)}}{\pi}\right)\right) \\
\log \delta &= \frac{\left(\log(q) + \log\left(\frac{\sqrt{2\ln\left(\frac{1}{\epsilon}\right)}}{\pi}\right)\right)^2}{4n\log(q)}.
\end{aligned}$$

□

5.9 Solving LWE with binary error via unique Shortest Vector Problem

We may solve LWE with binary error via Kannan's embedding technique [136]. The following analysis is analogous to Section 4.8.

Lemma 21. *Let an LWE instance with binary error be parameterised by n and q and assume that*

$$\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\} = q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}.$$

Any lattice reduction algorithm achieving log root-Hermite factor

$$\log \delta_0 = \frac{\left(\log q - \log\left(\tau\sqrt{2\pi e}\right)\right)^2}{4n\log q}$$

solves LWE with binary error via reduction to uSVP for some fixed $\tau \leq 1$.

Proof: Since $\mathbf{e}$ is binary and of length m, we must have $\|\mathbf{e}\|_2 \leq \sqrt{m}$. Following the argument in Section 4.8, we can embed the LWE instance into a lattice L with uSVP

structure such that

$$\frac{\lambda_2(L)}{\lambda_1(L)} = \frac{\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}}{\|\mathbf{e}\|_2} = \frac{q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}}{\|\mathbf{e}\|_2} \geq \frac{q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}}{\sqrt{m}}\,.$$

On the other hand we require

$$\frac{\lambda_2(L)}{\lambda_1(L)} \geq \tau\delta_0^m$$

and so it is sufficient if

$$\frac{q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}}{\sqrt{m}} \geq \tau\delta_0^m\,.$$

For a fixed δ_0 the optimal subdimension is $m = \sqrt{\frac{n\log q}{\log \delta_0}}$ and so we require:

$$\begin{aligned}
\delta_0^m &= \frac{q^{1-\frac{n}{m}}}{\tau\sqrt{2\pi e}} \\
m^2\log\delta_0 &= m(\log q - \log\left(\tau\sqrt{2\pi e}\right)) - n\log q \\
\frac{n\log q}{\log\delta_0}\log\delta_0 &= \sqrt{\frac{n\log q}{\log\delta_0}}\left(\log q - \log\left(\tau\sqrt{2\pi e}\right)\right) - n\log q \\
4n^2\log^2 q &= \frac{n\log q}{\log\delta_0}\left(\log q - \log\left(\tau\sqrt{2\pi e}\right)\right)^2 \\
\log\delta_0 &= \frac{\left(\log q - \log\left(\tau\sqrt{2\pi e}\right)\right)^2}{4n\log q}\,.
\end{aligned}$$

□

5.10 The hybrid attack for LWE with binary error

In this section we introduce the hybrid attack for LWE with binary error as presented in [50, 52]. This is a modification to the LWE setting of the hybrid lattice reduction and Meet-in-the-Middle attack of Howgrave-Graham [133] on NTRU [131]. We note that in a subsequent work, Göpfert *et al.* [117] present a generalised quantum version of the hybrid attack for LWE, in which it is no longer a requirement that the errors are binary. We do not discuss this subsequent work further and in particular, we assume that the errors are chosen uniformly at random in $\{0, 1\}$.

Howgrave-Graham gave the first theoretical analysis of the hybrid attack, and the analysis was improved firstly by Hirschhorn *et al.* [130] and then in [50, 52]. In

a subsequent work, Wunderer [214] identifies issues with, and refines, the analysis in [50, 52]. In light of this, in Section 5.11 we improve the comparison given in [50, 52] of the hybrid attack with other approaches for solving LWE with binary error.

It is assumed in [50, 52] that the LWE with binary error instance is in *normal form*; that is, the secret follows the error distribution. The later work of Wunderer [214] removes this requirement on the secret by using a different lattice. We follow [50, 52] and assume that the components of the secret are chosen uniformly at random in $\{0, 1\}$.

The hybrid attack on LWE with binary error as presented in [50, 52] builds on an idea of Bai and Galbraith [29], which is mentioned in Section 5.2. The idea is to guess the first r components of the secret vector and apply a lattice attack on the remaining problem. Howgrave-Graham's algorithm [133] involves a Meet-in-the-Middle component to speed up this guessing.

The hybrid attack is summarised as Algorithm 1 and proceeds as follows. Let

$$(\mathbf{A}, \mathbf{c} = \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e} \mod q)$$

with $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $\mathbf{c} \in \mathbb{Z}_q^m$, $\tilde{\mathbf{s}} \in \{0, 1\}^n$ and $\mathbf{e} \in \{0, 1\}^m$ be an LWE instance with binary error $\mathbf{e}$ and binary secret $\tilde{\mathbf{s}}$. To obtain a smaller error vector we subtract the vector $\frac{1}{2} \cdot \mathbf{1}$ from $\mathbf{c}$. This yields a new LWE instance $(\mathbf{A}, \mathbf{c}' = \mathbf{A}\tilde{\mathbf{s}} + \mathbf{e}' \mod q)$, where $\mathbf{c}' = \mathbf{c} - \frac{1}{2} \cdot \mathbf{1}$ and $\mathbf{e}' = \mathbf{e} - \frac{1}{2} \cdot \mathbf{1}$. In particular, we expect the original error vector $\mathbf{e}$ to have $\frac{m}{2}$ nonzero entries $\mathbf{e}_{(i)} = 1$ so its expected norm is $\frac{\sqrt{m}}{\sqrt{2}}$. The new error vector $\mathbf{e}'$ has m nonzero entries $\mathbf{e}'_{(i)}$ each of which satisfies $\|\mathbf{e}'_{(i)}\| = \frac{1}{2}$, so $\|\mathbf{e}'\|_2 = \frac{\sqrt{m}}{2}$.

For $r \in \{1, \ldots, n-1\}$, let $\mathbf{v} \in \{0, 1\}^r$, $\mathbf{s} \in \{0, 1\}^{n-r}$ be such that $\tilde{\mathbf{s}} = \begin{pmatrix} \mathbf{v} \\ \mathbf{s} \end{pmatrix}$; and let $\mathbf{A}_1 \in \mathbb{Z}_q^{m \times r}$ and $\mathbf{A}_2 \in \mathbb{Z}_q^{m \times (n-r)}$ be such that $\mathbf{A} = (\mathbf{A}_1 | \mathbf{A}_2)$. We can rewrite the LWE instance $(\mathbf{A}, \mathbf{c}')$ as

$$\mathbf{c}' = (\mathbf{A}_1 | \mathbf{A}_2) \begin{pmatrix} \mathbf{v} \\ \mathbf{s} \end{pmatrix} + \mathbf{e}' = \mathbf{A}_1\mathbf{v} + \mathbf{A}_2\mathbf{s} + \mathbf{e}' \mod q\,.$$

The main idea of the hybrid attack is to guess $\mathbf{v}$ and solve the remaining LWE instance $(\mathbf{A}_2, \tilde{\mathbf{c}} = \mathbf{c}' - \mathbf{A}_1\mathbf{v} = \mathbf{A}_2\mathbf{s} + \mathbf{e}' \mod q)$, which has binary secret $\mathbf{s}$ and error $\mathbf{e}' \in \{-\frac{1}{2}, \frac{1}{2}\}^m$. The newly obtained LWE instance is solved using a decoding

approach in the lattice $\Lambda_q(\mathbf{A}_2)$, as discussed in Section 4.7. To briefly recall, since $\tilde{\mathbf{c}} = \mathbf{A}_2\mathbf{s} + q\mathbf{w} + \mathbf{e}'$ for some vector $\mathbf{w} \in \mathbb{Z}^m$ and $\mathbf{e}'$ is small, $\tilde{\mathbf{c}}$ is close to the lattice vector $\mathbf{A}_2\mathbf{s} + q\mathbf{w} \in \Lambda_q(\mathbf{A}_2)$. Hence we can hope to find $\mathbf{e}'$ by, for example, running Babai's Nearest Plane algorithm [27] in combination with a sufficient lattice basis reduction as a precomputation. Recall (Section 2.2) that we denote by $\mathrm{NP}_{\mathbf{B}}(\mathbf{t})$ the output of Babai's Nearest Plane on input a basis $\mathbf{B}$ and a target vector $\mathbf{t}$. If the basis is clear from the context, we omit it from the notation and write $\mathrm{NP}(\mathbf{t})$.

The quality of the basis obtained from the precomputation is, as usual, characterised by the root-Hermite factor δ_0. Running a stronger lattice reduction algorithm would mean more time is taken for the precomputation but the quality of the basis $\mathbf{B}$ of the lattice $\Lambda_q(\mathbf{A}_2)$ would increase and hence the running time of the hybrid attack would decrease. The value δ_0 therefore determines the trade-off between the runtime of the precomputation and the actual hybrid attack, and should be optimised.

As well as optimising the choice of δ_0 we need to optimise r, the guessing dimension. On the one hand, increasing r increases the complexity of guessing part, since more entries of the secret have to be guessed. On the other hand, increasing r increases the determinant of the lattice, making the decoding part of the attack easier. Since there are only finitely many choices for r, the optimal choice can be found numerically: for each r, calculate the optimal δ_0 that minimises the runtime; then choose the r such that the overall runtime is minimised.

The guessing of $\mathbf{v}$ is sped up by a Meet-in-the-Middle approach; that is, by guessing binary vectors $\mathbf{v}_1 \in \{0,1\}^r$ and $\mathbf{v}_2 \in \{0,1\}^r$ such that $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2$. In order for this to make sense, we need to ensure two things. Firstly, we note that without performing a Meet-in-the-Middle improvement to the guessing step, and assuming a sufficiently good input basis $\mathbf{B}$, we have

$$\mathrm{NP}_{\mathbf{B}}(\tilde{\mathbf{c}}) = \mathrm{NP}_{\mathbf{B}}(\mathbf{c}' - \mathbf{A}_1\mathbf{v}) = \mathbf{e}'\,.$$

We therefore need that

$$\mathrm{NP}_{\mathbf{B}}(\mathbf{c}' - \mathbf{A}_1\mathbf{v}) = \mathrm{NP}_{\mathbf{B}}(-\mathbf{A}_1\mathbf{v}_1) + \mathrm{NP}_{\mathbf{B}}(\mathbf{c}' - \mathbf{A}_1\mathbf{v}_2)$$

holds for $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2$. Secondly, we need ensure that we store guesses $\mathbf{v}_1$ in such a way that we can recognise pairs $\mathbf{v}_1$ and $\mathbf{v}_2$ for which the above holds.

Algorithm 1: The hybrid attack as presented in [50, 52].

Input: $q, r \in \mathbb{Z}$
$\mathbf{A} = (\mathbf{A}_1 | \mathbf{A}_2)$, where $\mathbf{A}_1 \in \mathbb{Z}_q^{m \times r}, \mathbf{A}_2 \in \mathbb{Z}_q^{m \times (n-r)}$
$\mathbf{c} \in \mathbb{Z}_q^m$
$\mathbf{B}$, a lattice basis of $\Lambda_q(\mathbf{A}_2)$

calculate $c = \lfloor r/4 \rceil$;
calculate $\mathbf{c}' = \mathbf{c} - \left(\frac{1}{2}\right) \cdot \mathbf{1}$;
while *true* **do**
  guess a binary vector $\mathbf{v}_1 \in \{0,1\}^r$ with c 1s ;
  calculate $\mathbf{x}_1 = -\operatorname{NP}_{\mathbf{B}}(-\mathbf{A}_1\mathbf{v}_1) \in \mathbb{R}^m$;
  calculate $\mathbf{x}_2 = \operatorname{NP}_{\mathbf{B}}(\mathbf{c}' - \mathbf{A}_1\mathbf{v}_1) \in \mathbb{R}^m$;
  store $\mathbf{v}_1$ in all the boxes addressed by $\mathcal{A}_{\mathbf{x}_1}^{(r)} \cup \mathcal{A}_{\mathbf{x}_2}^{(r)}$;
  for *all* $\mathbf{v}_2 \neq \mathbf{v}_1$ *in all the boxes addressed by* $\mathcal{A}_{\mathbf{x}_1}^{(r)} \cup \mathcal{A}_{\mathbf{x}_2}^{(r)}$ **do**
    Set $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2$ and calculate $\mathbf{x} = \left(\frac{1}{2}\right) \cdot \mathbf{1} + \operatorname{NP}_{\mathbf{B}}(\mathbf{c}' - \mathbf{A}_1\mathbf{v}) \in \mathbb{R}^m$;
    if $\mathbf{x} \in \{0,1\}^m$ *and* $\exists\, \tilde{\mathbf{s}} \in \{0,1\}^n : \mathbf{c} = \mathbf{A}\tilde{\mathbf{s}} + \mathbf{x} \mod q$ **then**
      return $\mathbf{x}$;

Let $\mathbf{t}_1 = -\mathbf{A}_1\mathbf{v}_1$ and $\mathbf{t}_2 = \mathbf{c}' - \mathbf{A}_1\mathbf{v}_2$. Then we can restate the first requirement as

$$\operatorname{NP}_{\mathbf{B}}(\mathbf{t}_1) + \operatorname{NP}_{\mathbf{B}}(\mathbf{t}_2) = \operatorname{NP}_{\mathbf{B}}(\mathbf{t}_1 + \mathbf{t}_2) = \mathbf{e}';$$

that is,

$$\operatorname{NP}_{\mathbf{B}}(\mathbf{t}_2) - \mathbf{e}' = -\operatorname{NP}_{\mathbf{B}}(\mathbf{t}_1)\,.$$

Let also $\mathbf{x}_1 = -\operatorname{NP}_{\mathbf{B}}(\mathbf{t}_1)$ and $\mathbf{x}_2 = \operatorname{NP}_{\mathbf{B}}(\mathbf{t}_2)$. Then we see that $\mathbf{x}_1 = \mathbf{x}_2 - \mathbf{e}'$; that is, $\mathbf{x}_1$ and $\mathbf{x}_2$ differ by $\pm\frac{1}{2}$ on each component since $\mathbf{e}' \in \{-\frac{1}{2}, \frac{1}{2}\}^m$.

To satisfy the second requirement in Algorithm 1 we store a guess $\mathbf{v}_1$ in hash boxes addressed according to $\mathbf{x}_1$ and $\mathbf{x}_2$ as specified in Definition 18.

Definition 18. *Let $m \in \mathbb{N}$. For a vector $\mathbf{x} \in \mathbb{R}^m$ the set $\mathcal{A}_{\mathbf{x}}^{(m)} \subset \{0,1\}^m$ is defined as*

$$\mathcal{A}_{\mathbf{x}}^{(m)} = \left\{ \mathbf{z} \in \{0,1\}^m \;\middle|\; \begin{array}{l} \mathbf{z}_{(i)} = 1 \text{ for all } i \in \{1,\ldots,m\} \text{ with } \mathbf{x}_{(i)} > -1/2\,, \text{ and} \\ \mathbf{z}_{(i)} = 0 \text{ for all } i \in \{1,\ldots,m\} \text{ with } \mathbf{x}_{(i)} < -1/2 \end{array} \right\}.$$

We now give the intuition for this definition, showing that pairs $\mathbf{v}_1$ and $\mathbf{v}_2$ will be recognised. For $\mathbf{x}_1$ obtained during Algorithm 1, the set $\mathcal{A}_{\mathbf{x}_1}^{(m)}$ consists of the sign vector of $\mathbf{x}_1$, where 1 represents a nonnegative sign and 0 represents a negative sign. For $\mathbf{x}_2$ obtained during Algorithm 1, the set $\mathcal{A}_{\mathbf{x}_2}^{(m)}$ captures all the possible sign

vectors of $\mathbf{x}_2$ plus a vector in $\{-\frac{1}{2}, \frac{1}{2}\}^m$. Hence since $\mathbf{x}_1 = \mathbf{x}_2 - \mathbf{e}'$, valid pairs will collide in at least one box. The above discussion proves Lemma 22.

Lemma 22. *Let* $\mathbf{c}' = \mathbf{c} - \left(\frac{1}{2}\right) \cdot \mathbf{1}$ *and* $\mathbf{e}' = \mathbf{e} - \left(\frac{1}{2}\right) \cdot \mathbf{1}$. *Assume that* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *are guessed in separate loop iterations of Algorithm 1 and satisfy* $\mathbf{v}_1 + \mathbf{v}_2 = \mathbf{v}$. *Let* $\mathbf{t}_1 = -\mathbf{A}_1\mathbf{v}_1$ *and* $\mathbf{t}_2 = \mathbf{c}' - \mathbf{A}_1\mathbf{v}_2$ *and assume* $\mathrm{NP}(\mathbf{t}_1) + \mathrm{NP}(\mathbf{t}_2) = \mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2) = \mathbf{e}'$ *holds. Then* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *collide in at least one box chosen during Algorithm 1 and Algorithm 1 outputs the error vector* $\mathbf{e}$ *of the given LWE instance.*

We prove an equivalent requirement to $\mathrm{NP}(\mathbf{t}_1) + \mathrm{NP}(\mathbf{t}_2) = \mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2)$, which was an assumption in Lemma 22. We will need the following definition.

Definition 19. *Let* $m \in \mathbb{N}$. *A vector* $\mathbf{x} \in \mathbb{Z}^m$ *is called* $\mathbf{y}$-admissible *for some vector* $\mathbf{y} \in \mathbb{Z}^m$ *if* $\mathrm{NP}(\mathbf{x}) = \mathrm{NP}(\mathbf{x} - \mathbf{y}) + \mathbf{y}$.

The output of NP on input $\mathbf{x}$ is a vector $\mathbf{e}$ such that $\mathbf{x}-\mathbf{e}$ is a lattice point. Intuitively, Definition 19 captures this is the same lattice point for input $\mathbf{x}$ and input $\mathbf{x} - \mathbf{y}$. We can see this as follows. Let $\mathrm{NP}(\mathbf{x}) = \mathbf{e}_1$ and let $\mathbf{x} - \mathbf{e}_1 = \mathbf{z}_1 \in L$. Let also $\mathrm{NP}(\mathbf{x} - \mathbf{y}) = \mathbf{e}_2$ and let $\mathbf{x} - \mathbf{y} - \mathbf{e}_2 = \mathbf{z}_2 \in L$. If $\mathbf{x}$ is $\mathbf{y}$-admissible then:

$$\begin{aligned}
\mathrm{NP}(\mathbf{x}) &= \mathbf{y} + \mathrm{NP}(\mathbf{x} - \mathbf{y}) \\
-\mathrm{NP}(\mathbf{x}) &= -\mathbf{y} - \mathrm{NP}(\mathbf{x} - \mathbf{y}) \\
\mathbf{x} - \mathrm{NP}(\mathbf{x}) &= \mathbf{x} - \mathbf{y} - \mathrm{NP}(\mathbf{x} - \mathbf{y}) \\
\mathbf{x} - \mathbf{e}_1 &= \mathbf{x} - \mathbf{y} - \mathbf{e}_2 \\
\mathbf{z}_1 &= \mathbf{z}_2 \,.
\end{aligned}$$

Lemma 23. *Let* $\mathbf{t}_1 \in \mathbb{R}^m, \mathbf{t}_2 \in \mathbb{R}^m$ *be two arbitrary target vectors. Then the following are equivalent.*

(i) $\mathrm{NP}(\mathbf{t}_1) + \mathrm{NP}(\mathbf{t}_2) = \mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2)$.

(ii) $\mathbf{t}_1$ *is* $\mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2)$*-admissible.*

(iii) $\mathbf{t}_2$ *is* $\mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2)$*-admissible.*

Proof: By symmetry it suffices to show (i) if and only if (ii). We can write

$$\mathbf{y} = \mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2) = \mathrm{NP}(\mathbf{t})$$

where $\mathbf{t}$ is defined as $\mathbf{t} = \mathbf{t}_1 + \mathbf{t}_2$. Then (ii) is equivalent to the statement that $\mathbf{t}_1$ is $\mathbf{y}$-admissible, which by definition means $\mathrm{NP}(\mathbf{t}_1) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{y}) + \mathbf{y}$.

For all $\mathbf{x} \in \mathbb{R}^m$ and for all $\mathbf{z} \in L$, we have $\mathrm{NP}(\mathbf{x}) = \mathrm{NP}(\mathbf{x} - \mathbf{z})$. By definition, $\mathbf{t} - \mathbf{y} = \mathbf{t} - \mathrm{NP}(\mathbf{t}) \in L$. So,

$$\mathrm{NP}(\mathbf{t}_1 - \mathbf{y}) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{y} - (\mathbf{t} - \mathbf{y})) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{y} - \mathbf{t} + \mathbf{y}) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{t})\,.$$

Therefore we have

$$\mathrm{NP}(\mathbf{t}_1 - \mathbf{y}) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{t}) = \mathrm{NP}(\mathbf{t}_1 - (\mathbf{t}_1 + \mathbf{t}_2)) = \mathrm{NP}(-\mathbf{t}_2) = -\mathrm{NP}(\mathbf{t}_2)\,,$$

where the last equality is by symmetry. Using the above argument, we can conclude that (i) if and only if (ii) from the following:

$$\begin{aligned}
\mathrm{NP}(\mathbf{t}_1) + \mathrm{NP}(\mathbf{t}_2) &= \mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2) \\
\mathrm{NP}(\mathbf{t}_1) + \mathrm{NP}(\mathbf{t}_2) &= \mathbf{y} \\
\mathrm{NP}(\mathbf{t}_1) - \mathbf{y} &= -\mathrm{NP}(\mathbf{t}_2) \\
\mathrm{NP}(\mathbf{t}_1) - \mathbf{y} &= \mathrm{NP}(\mathbf{t}_1 - \mathbf{y}) \\
\mathrm{NP}(\mathbf{t}_1) &= \mathrm{NP}(\mathbf{t}_1 - \mathbf{y}) + \mathbf{y}\,.
\end{aligned}$$

□

We omit the remainder of the analysis of the hybrid attack given in [50, 52] but present in Theorem 10 the main result of that work. Theorem 10 gives the probability that Algorithm 1 terminates and its running time in the case that it does.

Theorem 10. *Let $c \in \mathbb{N}$ and let $r = 4c$. Let $(\mathbf{A}, \mathbf{c})$ be an LWE instance with binary error, where $\mathbf{A} = (\mathbf{A}_1 | \mathbf{A}_2)$ for $\mathbf{A}_1 \xleftarrow{\$} \mathbb{Z}_q^{m \times r}$ and $\mathbf{A}_2 \xleftarrow{\$} \mathbb{Z}_q^{m \times (n-r)}$. Let $\mathbf{B}$ be a basis of $\Lambda_q(\mathbf{A}_2)$ of quality given by root-Hermite factor δ_0. Let q, r, $\mathbf{A}$, $\mathbf{c}$, $\mathbf{B}$ be the input to Algorithm 1. Then, if Algorithm 1 terminates, it finds a valid binary error vector of the LWE with binary error instance $(\mathbf{A}, \mathbf{c})$. The probability that Algorithm 1 terminates is*

$$p_0 = 2^{-r} \binom{r}{2c} \prod_{i=1}^{m} \left(1 - \frac{2}{B(\frac{m-1}{2}, \frac{1}{2})} \int_{-1}^{\max(-r_i, -1)} (1 - t^2)^{\frac{m-3}{2}} dt\right),$$

where $B(\cdot, \cdot)$ denotes the Euler beta function [182] and

$$r_i = \frac{\delta_0^{-2(i-1)+m} q^{\frac{m-n+r}{m}}}{2\sqrt{m/4}}\,.$$

If Algorithm 1 terminates, the expected number of operations is

$$2^{16}\binom{r}{c}\left(p\binom{2c}{c}\right)^{-1/2},$$

where

$$p=\prod_{i=1}^{m}\left(1-\frac{1}{r_i B(\frac{m-1}{2},\frac{1}{2})}J(r_i,m)\right),$$

and

$$J(r_i,m)=\begin{cases}\int_{-r_i-1}^{r_i-1}\int_{-1}^{z+r_i}(1-y^2)^{\frac{m-3}{2}}dydz \\ \quad+\int_{r_i-1}^{-r_i}\int_{z-r_i}^{z+r_i}(1-y^2)^{\frac{m-3}{2}}dydz & \text{for } r_i<\frac{1}{2}\\ \int_{-r_i-1}^{-r_i}\int_{-1}^{z+r_i}(1-y^2)^{\frac{m-3}{2}}dydz & \text{for } r_i\geq\frac{1}{2}\,.\end{cases}$$

The value p referred to in Theorem 10 is the probability that $-\mathbf{A}_1\mathbf{v}_1$ is $\mathbf{e}'$-admissible for an appropriate guess $\mathbf{v}_1$. Howgrave-Graham [133] determines the probability p using experiments. A mathematical calculation of p is presented by Hirschhorn *et al.* [130], but this requires an additional assumption. The probability p is calculated analytically in [50, 52] without requiring this additional assumption.

Wunderer [214] identifies two issues with the analysis in [50, 52]. The first is that the runtime of lattice basis reduction is estimated according to Lindner and Peikert [150]. As we discussed in Section 4.6.1, Lindner and Peikert's estimate is inaccurate as it implies a subexponential algorithm for solving LWE. Wunderer's second argument is that Heuristic 2 needs to be modified for q-ary lattices, if the lattice reduction used as preprocessing is not sufficiently strong. Since BKZ 2.0 with blocksize 60 can now be easily run on a laptop [88], this issue can be reasonably ignored.

5.11 Comparison of the hybrid attack with other algorithms for LWE with binary error

In this section we compare the hybrid attack with other algorithms for solving LWE with binary error. This updates and extends the comparison presented in [50] and is also based on the subsequent work of Wunderer [214].

In Chapter 4 we saw that some algorithms for (general) LWE require a large number of samples m to be available in order to succeed. In the case of LWE with binary

5.11 Comparison of the hybrid attack with other algorithms for LWE with binary error

n	q	Hybrid attack (overestimate)	Hybrid attack (underestimate)	usvp	dual
128	256	47.9	26.1	58.2	53.5
160	256	58.6	35.2	74.3	60.0
192	256	65.8	44.8	91.2	73.1
224	256	75.3	54.4	109.0	86.8
256	256	85.4	64.1	127.0	101.0
288	256	94.4	79.3	145.8	115.8

Table 5.1: Comparison of approaches for solving LWE with binary error using at most $m = 2n$ samples. The bit hardness of the hybrid attack, reduction to uSVP and a distinguishing attack are given.

error, we saw in Section 5.6 that once the number of samples is at least quadratic in n, the problem can be solved in polynomial time. Furthermore, with slightly more than linearly many samples, such as $m = \mathcal{O}(n \log \log n)$, the algorithm given in [10] is subexponential. Therefore, it is argued in [50, 52] that it is not reasonable to deploy a scheme whose hardness is based on LWE with binary error if the attacker would have access to more than linearly many samples. In the analysis below, we assume the attacker has access to linearly many samples; we fix $m = 2n$ for concreteness. We note that since Kirchner and Fouque [139] showed that LWE with binary error can be solved in subexponential time using only $m = n$ samples this may itself not be a reasonable assumption.

We now compare the hybrid attack with alternative approaches for solving LWE with binary error. As mentioned in Section 5.10, the comparison in [50, 52] uses Lindner and Peikert [150] to estimate the cost of lattice reduction, which is inaccurate. We will estimate lattice reduction differently; we specify how below.

We compare the hybrid attack with distinguishing via lattice reduction, as described in Section 5.8; and reduction to uSVP, as described in Section 5.9. Following [50, 52] we do not consider Arora and Ge, Meet-in-the-Middle or BKW variants. We also omit from consideration a decoding approach analogous to that described in Section 4.7. This was considered in detail in [50, 52] but we were unable to update the code used to generate the estimate for the cost of this approach. The results of the comparison are presented in Table 5.1.

To estimate the cost of the hybrid attack we update the work of Wunderer [215]. In

particular, Wunderer gives both an overestimate and an underestimate of the cost of the hybrid attack, using the following methodology for the cost of lattice reduction. Given a target δ_0 specifying the quality of lattice reduction, the minimal necessary blocksize k is determined using Chen's thesis [64]. For the underestimate, it is assumed that one round of BKZ with blocksize k is sufficient. For the overestimate, the BKZ 2.0 simulator [65] is used to determine the number of rounds needed. It is assumed that the SVP oracle is implemented as enumeration.

For our hybrid attack estimates, we take a slightly different approach. We again determine the minimal necessary blocksize k, given the δ_0, using Chen's thesis [64]. For the underestimate, we assume that lattice reduction costs one call to the SVP oracle and the SVP oracle is implemented as sieving, as in Alkim *et al.* [19]. For the overestimate, we assume the cost of lattice reduction is the same as is given in commit f13c4a7 of the LWE estimator [6]. In particular we assume that there are $8n$ calls to the SVP oracle when calling BKZ in dimension n with blocksize k, and that the SVP oracle is implemented using sieving as in Becker *et al.* [33].

For both the distinguishing approach, and reduction to uSVP, we also need to estimate the cost of lattice reduction. To obtain estimates for these approaches, we again assume lattice reduction costs the same as that given in commit f13c4a7 of the LWE estimator.

Since we fix $m = 2n$ samples, it is possible that we do not have enough samples to use the optimal subdimension $m = \sqrt{\frac{n \log q}{\log \delta_0}}$ when estimating the uSVP or distinguishing approaches. To determine whether or not we have enough samples, we firstly calculate δ_0 assuming we have as many samples as we need for the optimal subdimension, as specified in Lemma 21 and Lemma 20 respectively. Then we check that the corresponding m is indeed less than or equal to $2n$. If so, we use the runtime estimate for that δ_0. If not, we take $m = 2n$ and derive the corresponding

root-Hermite factor, and use the runtime estimate for that δ_0. For uSVP we obtain:

$$\begin{aligned}\frac{q^{1-\frac{n}{m}}}{\sqrt{2\pi e}} &= \tau\delta_0^m \\ \frac{q^{1-\frac{n}{2n}}}{\sqrt{2\pi e}} &= \tau\delta_0^{2n} \\ \delta_0 &= \left(\frac{\sqrt{q}}{\tau\sqrt{2\pi e}}\right)^{\frac{1}{2n}}.\end{aligned}$$

For distinguishing we obtain:

$$\begin{aligned}\delta_0^m q^{\frac{n}{m}} &= q\frac{\sqrt{2\ln\left(\frac{1}{\epsilon}\right)}}{\pi} \\ \delta_0^{2n} q^{\frac{n}{2n}} &= q\frac{\sqrt{2\ln\left(\frac{1}{\epsilon}\right)}}{\pi} \\ \delta_0 &= \left(\frac{\sqrt{2q\ln\left(\frac{1}{\epsilon}\right)}}{\pi}\right)^{\frac{1}{2n}}.\end{aligned}$$

In Figure 5.1 we compare the cost of the hybrid attack with the best performing other algorithm, according to Table 5.1. For reference we also give the estimates for the cost of the hybrid attack for these parameters as given in [50]. It can be seen that the hybrid attack performs favourably compared with other approaches, and outperforms the best other approach for sufficiently large n.

We can alternatively compare the hybrid attack with other algorithms using the LWE estimator [6] directly. This also allows us to compare the cost of the hybrid attack with a decoding approach. We have done so using commit f13c4a7 of the LWE estimator and the results are given in Table 5.2 and Table 5.3.

The LWE estimator expects a discrete Gaussian error distribution, rather than a uniform binary error distribution, as we have here. In order to use the LWE estimator we need to appropriately model the error distribution as a discrete Gaussian. We do so using Heuristic 3 so that $\alpha = \frac{\sqrt{2\pi}}{2q}$. We use the limited sample functionality implemented by Bindel *et al.* [36] and call the LWE estimator with $m = 2n$ samples. Binary secrets are not currently supported, so we cannot indicate directly to the LWE estimator that there is a binary secret. However, ternary secrets, where the $\mathbf{s}_{(i)}$ are uniformly chosen from $\{-1, 0, 1\}$, are supported, and there is an option to specify the number of nonzero components h of the secret.

5.11 Comparison of the hybrid attack with other algorithms for LWE with binary error

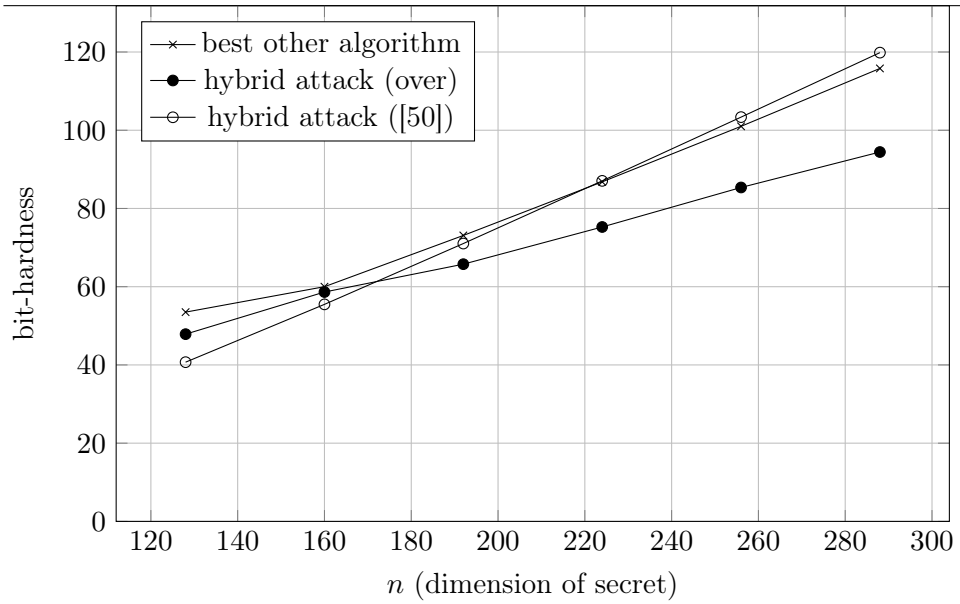

Figure 5.1: Estimates of solving LWE with binary error instances with $m = 2n$ samples and modulus $q = 256$ using the hybrid attack and the best other algorithm.

In Table 5.2 we call the LWE estimator without giving any information about the secret, whereas in Table 5.3 we model the binary secret as a ternary secret with $h = \frac{n}{2}$. This gives a secret that has the same number of nonzero components as the binary secret, and the components of the secret have the same norm as a binary secret. The column labelled dec in Table 5.3 was obtained by fixing the success probability to be 0.99, reflecting the behaviour of the LWE estimator in the later commit 6f25340. In commit f13c4a7, the decoding approach for sparse small secret was not well supported.

When analysing Table 5.3, we caution that the LWE estimator interprets a choice of h as the declaration of a sparse secret. It would then assume, for example, that the best performing distinguishing attack via lattice reduction is Albrecht's variant for small and sparse secret [7], whereas this may not be the case for a uniform binary secret. Similar assumptions are made in the case of the decoding and uSVP approaches.

Since we would expect a sparse secret instance to be easier to solve than a dense secret, in practice it may be harder to solve the LWE with binary error instances than it would appear from the results in Table 5.3. In contrast, it is likely that it in practice is easier to solve the LWE with binary error instances than it would appear from

5.11 Comparison of the hybrid attack with other algorithms for LWE with binary error

n	q	Hybrid attack (over)	Hybrid attack (under)	dec	usvp	dual
128	256	47.9	26.1	52.1	48.2	71.4
160	256	58.6	35.2	61.8	64.0	78.0
192	256	65.8	44.8	73.2	67.5	91.3
224	256	75.3	54.4	85.1	78.8	99.3
256	256	85.4	64.1	97.4	90.4	100.8
288	256	94.4	79.3	109.9	102.2	111.6

Table 5.2: Comparison of attacks on LWE with binary error obtained from the LWE estimator [16, 6] using $m = 2n$ samples and error distribution characterised by $\alpha = \sqrt{2\pi}/2q$.

n	q	Hybrid attack (over)	Hybrid attack (under)	dec	usvp	dual
128	256	47.9	26.1	45.4	53.2	56.0
160	256	58.6	35.2	60.1	64.7	65.8
192	256	65.8	44.8	62.0	74.4	76.9
224	256	75.3	54.4	71.4	85.0	87.9
256	256	85.4	64.1	81.2	96.5	99.0
288	256	94.4	79.3	91.2	108.3	109.9

Table 5.3: Comparison of attacks on LWE with binary error obtained from the LWE estimator [16, 6] using $m = 2n$ samples, error distribution characterised by $\alpha = \sqrt{2\pi}/2q$, and ternary secret with $h = n/2$.

Table 5.2, which assumes a general secret, rather than a small secret. Assuming that the decoding, uSVP and dual approaches cost somewhere in between the values given in Tables 5.2 and 5.3, we see that the hybrid attack performs favourably compared to the other approaches.

Chapter 6

Ring Learning with Errors

Contents

This chapter considers the Ring Learning with Errors problem. Most of this chapter is background material following the presentation in [173], which in turn follows Lyubashevsky et al. *[157, 158, 159, 160]. We also clarify a result due to Ducas and Durmus [95]: this is based on joint (unpublished) work with Cid and Murphy.*

The *Ring Learning with Errors* (Ring-LWE) problem, introduced by Lyubashevsky *et al.* [157], is a generalisation of the Learning with Errors problem from the ring of integers to other rings. Specifically, the problem is defined in the ring of integers R of a number field K. In applications, R is typically the m^{th} cyclotomic ring, and very often m is chosen as a power of 2.

In practice, Ring-LWE is often preferred to LWE for efficiency reasons. For example, key sizes in LWE are typically quadratic in the dimension n whereas in Ring-LWE they are linear [196]. The homomorphic encryption scheme [100] that we will consider in Chapter 7, as well as several others [47, 44, 160], are based on Ring-LWE. The hardness of Ring-LWE is the assumption used in a wealth of other applications besides homomorphic encryption [187], such as Attribute-based Encryption [84] and

obfuscation [87]. A popular application, considered in [15, 19, 40, 186, 90, 89, 150], is Ring-LWE based key exchange.

In Section 6.4 we discuss the hardness of Ring-LWE and related problems on ideal lattices. There is a reduction from presumed hard problems on ideal lattices to Ring-LWE [157, 189], assuming an appropriate error distribution is chosen. We therefore discuss various methods proposed in the literature to ensure the error distribution is chosen appropriately in Section 6.5. One such proposal is due to Ducas and Durmus [95] and we revisit their result [95, Theorem 5] in Section 6.6.

6.1 Algebraic background

Rings of integers of number fields. The minimal polynomial f of an algebraic number β is the unique irreducible monic polynomial with rational coefficients of smallest degree d such that $f(\beta) = 0$ and whose leading coefficient is 1. A number field $K = \mathbb{Q}(\beta) \cong \mathbb{Q}[x]/(f(x))$ is an extension of $\mathbb{Q}$ formed by adjoining β. Its ring of integers $R = \mathcal{O}_K$ is the ring of all elements of K that are roots of a monic polynomial with coefficients in $\mathbb{Z}$. Each root β_k of f defines an embedding $\sigma_k : K \to \mathbb{C}$. The *canonical embedding*

$$\sigma : \mathbb{Q}(\beta) \to \mathbb{C}^d$$

is the direct sum of these embeddings, so that $\sigma(z) = (\sigma_k(z))_{k\in\{1,\dots,d\}}$.

The ring of integers R embeds under σ as a lattice. The conjugate dual of this lattice corresponds to the embedding of the *dual fractional ideal*:

$$R^\vee = \{a \in K \mid \mathrm{Tr}(aR) \subset \mathbb{Z}\}.$$

We denote by $(R^\vee)^k$ the space of products of k elements of $R^\vee$:

$$\left(R^\vee\right)^k = \left\{s_1 \dots s_k \mid s_1, \dots, s_k \in R^\vee\right\}.$$

Definition 20 ([158])**.** *Let R be the ring of integers of a number field K. Let $q \geq 2$ be an integer modulus. Let $R^\vee$ be the dual fractional ideal of R. Let $R_q = R/qR$ and $R_q^\vee = R^\vee/qR^\vee$. Let $K_\mathbb{R} = K \otimes_\mathbb{Q} \mathbb{R}$.*

Let χ be a distribution over $K_{\mathbb{R}}$. Let $s \in R_q^\vee$ be a secret. A sample from the Ring-LWE distribution *$A_{s,\chi}$ over $R_q \times K_{\mathbb{R}}/qR^\vee$ is generated by choosing $a \leftarrow R_q$ uniformly at random, choosing $e \leftarrow \chi$ and outputting*

$$(a, b = (a \cdot s)/q + e \mod qR^\vee).$$

Let Ψ be a family of distributions over $K_{\mathbb{R}}$. The Search Ring-LWE *problem is defined as follows: given access to arbitrarily many independent samples from $A_{s,\chi}$ for some arbitrary $s \in R_q^\vee$ and $\chi \in \Psi$, find s.*

Let Υ be a distribution over a family of error distributions, each over $K_{\mathbb{R}}$. The average-case Decision Ring-LWE *problem is to distinguish with non-negligible advantage between arbitrarily many independent samples from $A_{s,\chi}$ for a random choice of $(s, \chi) \leftarrow \mathcal{U}\left(R_q^\vee\right) \times \Upsilon$, and the same number of uniformly random samples from $R_q \times K_{\mathbb{R}}/qR^\vee$.*

Cyclotomic rings. Let ζ_m be a primitive m^{th} root of unity. Its minimal polynomial is the m^{th} cyclotomic polynomial $\Phi_m(x)$. The number field $\mathbb{Q}(\zeta_m)$ is the m^{th} cyclotomic field and its ring of integers is $R = \mathbb{Z}[\zeta_m] \cong \mathbb{Z}[x]/(\Phi_m(x))$. The roots of $\Phi_m(x)$ are ζ_m^i for $i \in \mathbb{Z}_m^*$, so there are $n = \varphi(m)$ of them. Therefore $\mathbb{Q}(\zeta_m)$ has n embeddings given by $\sigma_k(y) = y(\zeta_m^k)$ for $k \in \mathbb{Z}_m^*$. The dual fractional ideal of $\mathbb{Z}[\zeta_m]$ is $\mathbb{Z}[\zeta_m]^\vee = \frac{1}{\Phi_m'(\zeta_m)}\mathbb{Z}[\zeta_m]$.

The canonical embedding induces a geometry on K with ℓ_2-norm $\|\cdot\|_2$ and ℓ_∞-norm $\|\cdot\|_\infty$ given by the element's respective norm under σ: that is,

$$\|a\|_2 = \|\sigma(a)\|_2$$

and

$$\|a\|_\infty = \|\sigma(a)\|_\infty.$$

Prime cyclotomic rings. If m is a prime we have $n = \varphi(m) = m - 1$ and

$$\Phi_m(x) = x^n + \cdots + x + 1.$$

The ring embeddings σ_k are defined by $\zeta_m \mapsto \zeta_m^k$ for $1 \leq k \leq n$. Define t such that

$$t^{-1} = m^{-1}(1 - \zeta_m)\,,$$

then $R^\vee = \langle t^{-1} \rangle$ and $(R^\vee)^k = \left\{ t^{-k} r_1 \dots r_k \mid r_1, \dots, r_k \in R \right\}$. For $m \geq 3$ a prime we let $n' = \frac{1}{2}n = \frac{1}{2}(m-1)$.

Power-of-two cyclotomic rings. For $m = 2^{k+1}$ for some k we have $n = 2^k$ and $\Phi_m(x) = x^n + 1$. In this case R is an exact scaling of $R^\vee$ [95]. In particular, we have $\frac{m}{2} R^\vee = R$.

Bases. We now specify various $\mathbb{Z}$-bases of K, R and $R^\vee$ in the case when m is prime. More general definitions can be found in Lyubashevsky *et al.* [160].

Definition 21 ([160])**.** *The* Powerful basis $\overrightarrow{p}$ *of* $K = \mathbb{Q}(\zeta_m)$ *and* $R = \mathbb{Z}[\zeta_m]$ *is*

$$\left\{ \zeta_m^0, \zeta_m^1, \dots, \zeta_m^{n-1} \right\}\,.$$

Definition 22 ([160])**.** *The* Decoding basis $\overrightarrow{d}$ *of* $R^\vee$ *is*

$$\left\{ \tfrac{1}{m}(\zeta_m^0 - \zeta_m^n), \tfrac{1}{m}(\zeta_m^1 - \zeta_m^n), \dots, \tfrac{1}{m}(\zeta_m^{n-1} - \zeta_m^n) \right\}\,.$$

Definition 23 ([160])**.** *The* Scaled Decoding basis *of* $(R^\vee)^k$ *is*

$$m^{-(k-1)}\overrightarrow{d} = \left\{ \tfrac{1}{m^k}(\zeta_m^0 - \zeta_m^n), \tfrac{1}{m^k}(\zeta_m^1 - \zeta_m^n), \dots, \tfrac{1}{m^k}(\zeta_m^{n-1} - \zeta_m^n) \right\}\,.$$

6.2 The space H

The canonical embedding $\sigma : \mathbb{Q}(\beta) \to \mathbb{C}^d$ is the direct sum of the ring embeddings $\sigma_1, \dots, \sigma_d$. The ring embeddings $\sigma_1, \dots, \sigma_d$ occur in complex conjugate pairs with $\overline{\sigma_k} = \sigma_{m-k}$. Therefore σ maps into a subset of $\mathbb{C}^d$, which is denoted

$$H = \{ \mathbf{a} \in \mathbb{C}^d \quad | \quad \mathbf{a}_{(j)} = \overline{\mathbf{a}_{(m-j)}} \quad \forall j \in \{1, \dots, d\} \}\,.$$

Bases for H. Let $\mathbf{I}$ be the identity matrix and $\mathbf{J}$ the matrix obtained when the columns of $\mathbf{I}$ are reversed. In Definitions 25 and 26 we give two bases for H in the case that K is the m^{th} cyclotomic field for m a prime, so that $n = \varphi(m) = m - 1$.

We note that these bases can be defined more generally. The I-basis is the standard basis of H, expressing elements as a complex vector. The T-basis, given by the columns of the *conjugate pair matrix* $\mathbf{T}$, expresses an element of H as a real vector.

Definition 24 ([160])**.** *The* conjugate pair matrix $\mathbf{T}$ *is given by*

$$\mathbf{T} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 & i \\ 0 & 1 & \dots & 0 & 0 & \dots & i & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & i & \dots & 0 & 0 \\ 0 & 0 & \dots & 1 & -i & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \dots & 0 & 0 & \dots & -i & 0 \\ 1 & 0 & \dots & 0 & 0 & \dots & 0 & -i \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I} & i\mathbf{J} \\ \mathbf{J} & -i\mathbf{I} \end{pmatrix}.$$

Lemma 24 ([160])**.** *The conjugate pair matrix* $\mathbf{T}$ *is a unitary matrix with*

$$\mathbf{T}^{-1} = \mathbf{T}^{\dagger} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I} & \mathbf{J} \\ -i\mathbf{J} & i\mathbf{I} \end{pmatrix}.$$

Let $\mathbf{v}$ be a vector representing an element in the I-basis for H. Since $\mathbf{T}$ is unitary, we have $\|\mathbf{Tv}\|^2 = \|\mathbf{v}\|^2$; that is, the element has the same norm when represented in the T-basis. Expressing elements of H as vectors in the T-basis therefore gives an isomorphism between H and $\mathbb{R}^n$ as an inner product space [160].

Definition 25. *Let m be prime. The* I-basis *for H is given by the columns of the $n \times n$ identity matrix $\mathbf{I}$, that is to say by standard basis vectors. In particular the I-basis expresses an element in H as a vector of n' conjugate pairs.*

Definition 26. *Let m be prime. The* T-basis *for H is given by the columns of the conjugate pair matrix*

$$\mathbf{T} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I}_{n'} & i\mathbf{J}_{n'} \\ \mathbf{J}_{n'} & -i\mathbf{I}_{n'} \end{pmatrix}.$$

Embeddings of sums and products. The canonical embedding under σ of a sum in the cyclotomic number field gives a componentwise addition in H when elements

in H are expressed as vectors in any basis. Similarly, the canonical embedding under σ gives a componentwise multiplication when expressing the elements in H as vectors in the I-basis. We denote this componentwise multiplication as the $\odot$-*product* in H, and we have:

$$\begin{aligned}\sigma(aa') &= (\sigma_1(aa'), \ldots, \sigma_n(aa'))^T \\ &= (\sigma_1(a)\sigma_1(a'), \ldots, \sigma_n(a)\sigma_n(a'))^T \\ &= (\sigma_1(a), \ldots, \sigma_n(a))^T \odot (\sigma_1(a'), \ldots, \sigma_n(a'))^T \\ &= \sigma(a) \odot \sigma(a')\,.\end{aligned}$$

The canonical embedding of a product under σ gives other forms of product for the corresponding vectors expressing elements of H when other bases are used. When using the T-basis, the appropriate notion of a product of two elements of H, which are expressed as real vectors, is given by Definition 27. We define this multiplication as the $\otimes$-*product*.

Definition 27. *The* $\otimes$-product *of two real vectors* $\mathbf{u} = (u_{11}, u_{12}, \ldots, u_{n'1}, u_{n'2})^T$ *and* $\mathbf{v} = (v_{11}, v_{12}, \ldots, v_{n'1}, v_{n'2})^T$ *of length* $n = 2n'$ *is*

$$\mathbf{u} \otimes \mathbf{v} = \begin{pmatrix} u_{11} \\ u_{12} \\ \vdots \\ u_{n'1} \\ u_{n'2} \end{pmatrix} \otimes \begin{pmatrix} v_{11} \\ v_{12} \\ \vdots \\ v_{n'1} \\ v_{n'2} \end{pmatrix} = \mathbf{T}^\dagger (\mathbf{Tu} \odot \mathbf{Tv}) = 2^{-\frac{1}{2}} \begin{pmatrix} u_{11}v_{11} - u_{12}v_{12} \\ u_{11}v_{12} + u_{12}v_{11} \\ \vdots \\ u_{n'1}v_{n'1} - u_{n'2}v_{n'2} \\ u_{n'1}v_{n'2} + u_{n'2}v_{n'1} \end{pmatrix}.$$

6.3 Discretisation

Discretisation, in the context of Ring-LWE applications, is a process where a point in H is rounded to a nearby point in a lattice coset. Such a discretisation process usually involves randomisation, so gives rise to a random variable on the elements of the coset. In this section we describe the *coordinate-wise randomised rounding* method of discretisation [160, Section 2.4.2, first bullet point]. This section is based on [173, Section 5], and proofs for all results can be found there.

In Definition 28 we define the univariate Reduction random variable. This is a translation of the Bernoulli random variable [118], and so we can give its immediate

basic properties in Lemma 25. In particular we need to consider the multivariate generalisation of a Reduction random variable given by Definition 29.

Definition 28. *If Bern denotes the Bernoulli distribution, then the univariate* Reduction *distribution*

$$Red(a) = Bern(\lceil a \rceil - a) - (\lceil a \rceil - a)$$

is the discrete probability distribution defined for parameter $a \in \mathbb{R}$ as taking the values $1 + a - \lceil a \rceil$ with probability $\lceil a \rceil - a$ and $a - \lceil a \rceil$ with probability $1 - (\lceil a \rceil - a)$.

Lemma 25. *If $R_0 \sim Red(a)$ is a (univariate) Reduction random variable for parameter $a \in \mathbb{R}$, then R_0 satisifies (i) $|R_0| \leq 1$, (ii) $\mathrm{E}[R_0] = 0$, (iii) $\mathrm{Var}[R_0] \leq \frac{1}{4}$ and (iv) $a - R_0 \in \{\lfloor a \rfloor, \lceil a \rceil\} \subset \mathbb{Z}$.*

Definition 29. *A random variable $R = (R_1, \ldots, R_l)^T$ has a multivariate Reduction distribution $R \sim Red(\mathbf{a})$ on $\mathbb{R}^l$ for parameter $\mathbf{a} = (a_1, \ldots, a_l)^T \in \mathbb{R}^l$ if its components $R_j \sim Red(a_j)$ for $j = 1, \ldots, l$ are independent univariate Reduction random variables.*

We are now able to specify the coordinate-wise randomised rounding discretisation method and show this discretisation method is *valid*; that is, independent of the chosen coset representative.

Definition 30. *Suppose $\mathbf{B}$ is a (column) basis matrix for the n-dimensional lattice Λ in H. The* coordinate-wise randomised rounding discretisation $\lfloor X \rceil^{\mathbf{B}}_{\Lambda+\mathbf{c}}$ *of the random variable X to the lattice coset $\Lambda + \mathbf{c}$ with respect to the basis matrix $\mathbf{B}$ is then defined by the conditional random variable*

$$\left(\lfloor X \rceil^{\mathbf{B}}_{\Lambda+\mathbf{c}} \middle| X = \mathbf{x} \right) = \lfloor \mathbf{x} \rceil^{\mathbf{B}}_{\Lambda+\mathbf{c}} = \mathbf{x} + \mathbf{B}Q_{\mathbf{x},\mathbf{c}}$$

where $Q_{\mathbf{x},\mathbf{c}} \sim Red\left(\mathbf{B}^{-1}(\mathbf{c} - \mathbf{x})\right)$.

Lemma 26. *The coordinate-wise randomised rounding discretisation $\lfloor \mathbf{x} \rceil^{\mathbf{B}}_{\Lambda+\mathbf{c}}$ is a random variable on the lattice coset $\Lambda + \mathbf{c}$ and is valid.*

6.4 Hardness of Ring-LWE

As for the LWE problem, the Ring-LWE problem is related to well-studied lattice problems that are believed to be hard [189, 45, 158, 160, 185, 194]. Most notably

for cryptographic applications, Lyubashevsky *et al.* [157, 158] give a reduction from approximate SIVP on ideal lattices to Decision Ring-LWE when K is a cyclotomic number field. Recently, a much more general result, giving a reduction from worst-case lattice problems to Decision Ring-LWE for any number field, was presented by Peikert *et al.* [189]. If Decision Ring-LWE is hard, then a Ring-LWE sample looks pseudorandom. Hence we may argue, for example, indistinguishability of ciphertexts from random if the ciphertext is formed by masking the message with a Ring-LWE sample, as proposed by Lindner and Peikert [150].

Theorem 11 ([158])**.** *If K is a cyclotomic number field with ring of integers R, then there is a polynomial time quantum reduction from approximate SIVP on ideal lattices in K to Decision Ring-LWE in R given a fixed number of samples, where the error distribution is a fixed spherical Gaussian over H.*

Theorem 11 is based on two results. The first is a quantum reduction from a particular hard lattice problem to Search Ring-LWE in a ring of integers R of some number field K [158, Theorem 4.1]. The second specialises to the case of rings of integers of cyclotomic fields, giving a reduction from Search Ring-LWE to Decision Ring-LWE [158, Theorem 5.1]. This reduction from Search Ring-LWE to Decision Ring-LWE was generalised by Eisenträger *et al.* [97] to all rings of integers R of number fields K such that K is Galois and q splits completely in R. In turn, this reduction was further generalised by Chen *et al.* [62], who removed the requirement on q. A reduction from a particular hard lattice problem to Decision Ring-LWE for q of arbitrary shape had previously been given by Langlois and Stehlé [147].

The existence of a reduction, such as Theorem 11, from standard problems on ideal lattices gives more confidence that Ring-LWE is hard, since it shows that Ring-LWE is at least as hard as a well-studied problem on ideal lattices that is believed to be hard. However, there are at least two reasons why we might still be cautious. Firstly, as Chatterjee *et al.* [57] argue, the reduction may not be tight and hence it is unclear whether the concrete security assurance this would provide is meaningful. Secondly, the hardness of a problem on ideal lattices may be different than its analogue on general lattices: for example, the GapSVP problem is easy on ideal lattices [196]. Another example is the line of work [53, 79] which showed that finding short vectors in principal ideal lattices is potentially easier than for general lattices. Cramer *et al.* [80] generalised these results to show that worst-case approximate SVP on ideal lattices

with approximation factor $\gamma = \exp(\tilde{\mathcal{O}}(\sqrt{n}))$ can be solved in quantum polynomial time, whereas for a general lattice we would expect to reach $\gamma = \exp(\tilde{\Theta}(n))$.

Cramer *et al.* [80] stress that while Ring-LWE is at least as hard as approximate SVP on ideal lattices, it is not known to be equivalent, and hence their work should not currently impact on Ring-LWE-based schemes. Nonetheless, the inherent structure in the Ring-LWE setting has been seen by some as worrisome and rings with less structure have been proposed, for example by Bernstein *et al.* [35], as an alternative. The use of Module-LWE [147] as an alternative hardness assumption has been proposed [80], although Albrecht and Deo [11] showed that Module-LWE and Ring-LWE are polynomial-time equivalent.

On the other hand, we can consider Ring-LWE variants for which there is no longer a reduction from presumed hard problems on ideal lattices. Indeed in practice, some schemes based on Ring-LWE use an error distribution that may be too narrow for a reduction to apply. For example, the reductions would require an error distribution characterised by $\alpha q = \omega(\sqrt{\log n})$, where the standard deviation $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$. That is, αq is a function of n when the reduction holds. In contrast, in homomorphic encryption applications such as [121, 110, 58] we typically have $\alpha q = 8$; that is, αq is a constant. A number of works have given attacks on Ring-LWE variants when the underlying number field, error distribution, or modulus is of various special forms [56, 55, 62, 63, 97, 99].

6.5 Error distributions

Recall that in the hardness results for Ring-LWE in Section 6.4, in order for the reduction from ideal lattice problems to hold, we need to choose error polynomials e_i from a distribution which when embedded under the canonical embedding σ is a spherical Gaussian in H. So, we would sample the errors from a spherical Gaussian in H, then map them back using σ^{-1} to elements in K; indeed, strictly speaking [160], to elements in $K \otimes_{\mathbb{Q}} \mathbb{R}$.

A much more convenient, and arguably more natural, way to choose an error polynomial is such that each coefficient is chosen independently from a spherical Gaussian.

This gives rise to so-called *Polynomial-LWE* [210, 157, 47].

Definition 31 ([210, 157, 47])**.** *Let $f(x)$ be a monic irreducible polynomial in $\mathbb{Z}[x]$ of degree n. Let $R = \mathbb{Z}[x]/(f(x))$ and $R_q = \mathbb{Z}_q[x]/(f(x))$. Let $s \in R_q$ be a secret. Let $a_i \in R_q$ be chosen uniformly. Let e_i be chosen from the discrete Gaussian distribution on R centred at 0 and with standard deviation σ, spherical with respect to the power basis. Let $c_i = a_i s + e_i$.*
Decision Polynomial-LWE *is the problem of deciding whether pairs $(a_i, c_i) \in R_q \times R_q$ are sampled according to the above or are uniformly random.*
Search Polynomial-LWE *is the problem of recovering s given arbitrarily many samples of the form $(a_i, c_i) = (a_i, a_i s + e_i)$.*

The coefficient embedding map gives another map from K to $\mathbb{R}^n$, sending the error polynomial $e_i = e_{i,0} + e_{i,1}x + \cdots + e_{i,n-1}x^{n-1}$ to the vector $(e_{i,0}, e_{i,1}, \ldots, e_{i,n-1})$. The coefficient embedding and canonical embedding are related to one another by a linear transformation of $\mathbb{R}^n$ and in the case of the m^{th} cyclotomic for m a power of 2, this transformation is an isometry [95, 160]. This means that for a power-of-2 m, a spherical Gaussian distribution in the canonical embedding corresponds to a spherical distribution in the coefficient embedding, or equivalently, Polynomial-LWE and Ring-LWE are equivalent for power-of-2 cyclotomics. Hence we can choose the error polynomials in this more natural way and retain the hardness reduction. This is a key reason why in applications, designers of cryptosystems based on Ring-LWE favour the power-of-2 cyclotomic case, as in, for example, [47, 209, 216, 40]. This choice can also provide efficiency in implementations [160].

Ducas and Durmus [95] propose an alternative way to sample the errors more conveniently. They show that for any cyclotomic field K we can equally simply sample an error polynomial coefficient-wise in an appropriate extension of K according to a spherical Gaussian and perform a modular reduction to obtain an error in K, which is distributed as an elliptical Gaussian. When considered under σ, the errors retain the appropriate spherical Gaussian distribution in H. We revisit a key result of Ducas and Durmus in Section 6.6 below.

Another variant, used by Peikert *et al.* [20, 188, 81, 82] and equivalent to Ring-LWE, is *tweaked Ring-LWE*. In this variant a *tweak* factor $t \in R$ such that $tR^\vee = R$ is used. The secret s' is such that $s' = t \cdot s$, where s is a standard Ring-LWE secret, and

the errors e_i are chosen from a tweaked error distribution $t\psi$, where ψ is a spherical Gaussian as required in standard Ring-LWE.

6.6 The variant of Ducas and Durmus

In this section we review a result of Ducas and Durmus [95]. The work [95] is motivated by the fact that Ring-LWE can be complicated to implement for a cyclotomic ring when m is not a power of 2, since in such a setting the Ring-LWE definition in [157] involves $R^\vee$. Ducas and Durmus show that for any cyclotomic polynomial $\Phi_m(x)$, it is possible to avoid this requirement and always work in the ring $R = \mathbb{Z}[x]/(\Phi_m(x))$. In particular, their main result [95, Theorem 2] states the analogue of Theorem 11; that is, for cyclotomic rings there is a quantum reduction from approximate SVP on ideal lattices to a variant of Decision Ring-LWE, in which we must distinguish k samples $(a_i, a_i s + e_i) \in R_q \times R_q$ from k uniformly random samples, where $a_i, s \xleftarrow{\$} R_q$ and the e_i are chosen in a certain extension ring and then reduced modulo $\Phi_m(x)$.

The extension ring of $\mathbb{Q}[x]/(\Phi_m(x))$ is $\mathbb{Q}[x]/(\Theta_m(x))$, where $\Theta_m(x)$ is the polynomial that is $x^m - 1$ if m is odd, and $x^{m/2} + 1$ if m is even. A key component in the proof of the main result [95, Theorem 2] is the result [95, Theorem 5] showing that if the errors are chosen in $\mathbb{Q}[x]/(\Theta_m(x))$ in an appropriate way, this implies an appropriate spherical error distribution in H as required for hardness. However, the proof of [95, Theorem 5] contains a small error, although the statement is correct. Furthermore the notation used in [95] is sometimes non-standard. We revisit the proof of [95, Theorem 5], slightly altering the presentation, and show how the error can be corrected.

We begin by giving the relationships between the spaces $\mathbb{R}^{\varphi(m)} \cong H$, $\mathbb{Q}[x]/(\Theta_m(x))$ and $\mathbb{Q}[x]/(\Phi_m(x))$. These are summarised in Figure 6.1. The map

$$\beta : \mathbb{Q}[x]/(\Theta_m(x)) \to \mathbb{Q}[x]/(\Phi_m(x))$$

given by $x \mapsto x \mod \Phi_m(x)$ is reduction modulo Φ_m. The map

$$\sigma : \mathbb{Q}[x]/(\Phi_m(x)) \to H$$

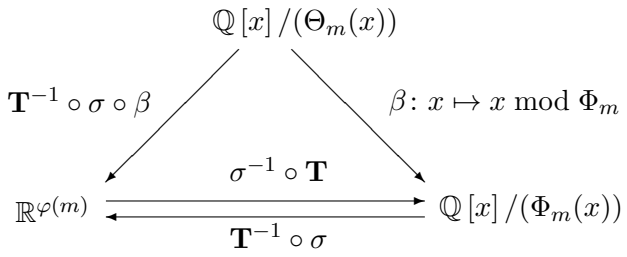

Figure 6.1: [95, Figure 1]. Mappings between different spaces.

is the canonical embedding of the number field $\mathbb{Q}[x]/(\Phi_m(x))$. We view H in the T-basis so that the map from H to $\mathbb{R}^{\varphi(m)}$ is given by the matrix $\mathbf{T}$ as specified in Definition 24. Let $\gamma : \mathbb{Z}[x]/(\Theta_m(x)) \to H \subseteq \mathbb{C}^{\varphi(m)}$ be the linear map from the power basis of $\mathbb{Z}[x]/(\Theta_m(x))$ to the canonical basis of $\mathbb{C}^{\varphi(m)}$; that is, the basis that extends the I-basis of H. Then γ is the composition map $\sigma \circ \beta$, as implied, though not explicitly stated, in [95]. Let $\mathbf{G}$ be the $\varphi(m) \times m'$ matrix representing γ. Then $\mathbf{T}^{-1} \circ \mathbf{G}$ gives the map from $\mathbb{Z}[x]/(\Theta_m(x))$ to $\mathbb{R}^{\varphi(m)}$.

Following the notation of [95], we let ψ_s be the spherical Gaussian with mean 0 and standard deviation s over $\mathbb{R}$ and ψ_s^d be the spherical Gaussian over $\mathbb{R}^d$ where each component is chosen independently from ψ_s. We let $m' = m$ if m is odd and $m' = \frac{m}{2}$ if m is even.

We now describe the change in our presentation compared to [95]. Ducas and Durmus [95] give the mapping from H to $\mathbb{R}^{\varphi(m)}$ by the matrix $\mathbf{T}_{DD} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I} & i\mathbf{I} \\ \mathbf{I} & -i\mathbf{I} \end{pmatrix}$, whereas we give this mapping by the matrix $\mathbf{T}$. The ordering of components of σ does not seem consistent when using the matrix $\mathbf{T}_{DD}$. For example, when $m = 5$ the matrix $\mathbf{T}_{DD}$ corresponds to the conjugates in the order $(\sigma_1, \sigma_2, \sigma_4, \sigma_3)$ whereas by definition, σ considers the canonical order $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$. Using the conjugate pair matrix $\mathbf{T}$ and the T-basis for H enables us to always use the canonical order for σ.

We now discuss and correct the proof presented by Ducas and Durmus of their result [95, Theorem 5]. We begin by presenting the theorem as stated in [95].

Theorem 12 ([95])**.** *Let $v \in \mathbb{Q}[x]/(\Theta_m(x))$ be a random variable distributed as $\psi_s^{m'}$ in the power basis. Then the distribution of $(\mathbf{T}_{DD}^{-1} \circ \sigma \circ \beta)(v)$, seen in the I-basis of*

H, is the spherical Gaussian $\psi_{s\sqrt{m'}}^{\varphi(m)}$.

Define the matrix $\mathbf{E}_{DD}$ as $\mathbf{E}_{DD} = \mathbf{T}_{DD}^{-1} \cdot \mathbf{G}$. The matrix $\mathbf{E}_{DD}$ represents the embedding directly from $\mathbb{Q}[X]/(\Theta_m(x))$ to $\mathbb{R}^{\varphi(m)}$. Let $\mathbf{I}_{\varphi(m)}$ be the $\varphi(m) \times \varphi(m)$ identity matrix.

The stated proof requires the following three claims:

(i) $\mathbf{G} \cdot \overline{\mathbf{G}^T} = m'\mathbf{I}_{\varphi(m)}$

(ii) $\mathbf{T}_{DD}^{-1}$ is Hermitian

(iii) $\mathbf{E}_{DD} = \overline{\mathbf{E}_{DD}}$, or, equivalently, $\mathbf{E}_{DD}$ is real.

Claim (ii) does not specify the property of $\mathbf{T}_{DD}$ that we would require. In particular, the claim that $\mathbf{T}_{DD}^{-1}$ is Hermitian would mean $\overline{(\mathbf{T}_{DD}^{-1})^T} = \mathbf{T}_{DD}^{-1}$. However, as we will see below, the desired relation is of the form $\overline{(\mathbf{T}_{DD}^{-1})^T} = \mathbf{T}_{DD}$.

In light of our change to the matrix $\mathbf{T}$, and noting the issue with the stated proof, we now restate and prove [95, Theorem 5].

Theorem 13 ([95])**.** *Let $v \in \mathbb{Q}[x]/(\Theta_m(x))$ be a random variable distributed as $\psi_s^{m'}$ in the power basis. Then the distribution of $(\mathbf{T}^{-1} \circ \sigma \circ \beta)(v)$, seen in the I-basis of H, is the spherical Gaussian $\psi_{s\sqrt{m'}}^{\varphi(m)}$.*

Proof: We follow the framework of the proof stated in [95]. Define the matrices $\mathbf{T}$, $\mathbf{G}$ as above and define the matrix representing the embedding directly from $\mathbb{Q}[x]/(\Theta_m(x))$ to $\mathbb{R}^{\varphi(m)}$ as $\mathbf{E} = \mathbf{T}^{-1} \cdot \mathbf{G}$.

Suppose the following claims are true:

(i) $\mathbf{G} \cdot \overline{\mathbf{G}^T} = m' \cdot \mathbf{I}_{\varphi(m)}$,

(ii) $\overline{(\mathbf{T}^{-1})^T} = \mathbf{T}$,

(iii) $\mathbf{E} = \overline{\mathbf{E}}$.

In this case, the following argument holds:

$$\begin{aligned}
\mathbf{E} \cdot \mathbf{E}^T &= \mathbf{E} \cdot \overline{\mathbf{E}}^T \\
&= \mathbf{T}^{-1} \cdot \mathbf{G} \cdot \overline{(\mathbf{T}^{-1} \cdot \mathbf{G})^T} \\
&= \mathbf{T}^{-1} \cdot \mathbf{G} \cdot \overline{\mathbf{G}^T} \cdot \overline{(\mathbf{T}^{-1})^T} \\
&= \mathbf{T}^{-1} \cdot \mathbf{G} \cdot \overline{\mathbf{G}^T} \cdot \mathbf{T} \\
&= \mathbf{T}^{-1} \cdot m' \cdot \mathbf{I}_{\varphi(m)} \cdot \mathbf{T} \\
&= m' \cdot \mathbf{I}_{\varphi(m)} \,,
\end{aligned}$$

giving the required result.

Claim (i) and Claim (iii) hold directly as in [95] For Claim (ii), recall $\mathbf{T} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I} & i\mathbf{J} \\ \mathbf{J} & -i\mathbf{I} \end{pmatrix}$. Therefore

$$\begin{aligned}
\mathbf{T}^{-1} &= \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I} & \mathbf{J} \\ -i\mathbf{J} & i\mathbf{I} \end{pmatrix} \\
(\mathbf{T}^{-1})^T &= \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I} & -i\mathbf{J} \\ \mathbf{J} & i\mathbf{I} \end{pmatrix} \\
\overline{(\mathbf{T}^{-1})^T} &= \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I} & i\mathbf{J} \\ \mathbf{J} & -i\mathbf{I} \end{pmatrix}
\end{aligned}$$

as required. □

6.6.1 The case m is an odd prime

In this section, we consider the case that m is an odd prime, as an extended example. We firstly establish notation in this case, and give details of matrices representing each of the maps between the various spaces. We then give more detail on the proof of claim (i), which was required in the proof given in the previous section of the result of Ducas and Durmus [95, Theorem 5]. We conclude with an alternative proof of [95, Theorem 5] restricted to the case that m is an odd prime.

In the case m is an odd prime, we have $\varphi(m) = m - 1$ and $\Phi_m(x) = x^{m-1} + \ldots + 1$. The extension ring is $\mathbb{Q}[x]/(\Theta_m(x))$, where $\Theta_m(x) = x^m - 1$. Let $m_1 = m - 1$ and $m_2 = \frac{1}{2}m_1 = \frac{1}{2}(m-1)$. Let $\mathbf{I}_1$ be the $m_1 \times m_1$ identity matrix, let $\mathbf{J}_1$ be the $m_1 \times m_1$ matrix with 1s on the reverse diagonal and 0 elsewhere, and let $\mathbf{K}_1$ be the $m_1 \times m_1$ matrix with every entry 1.

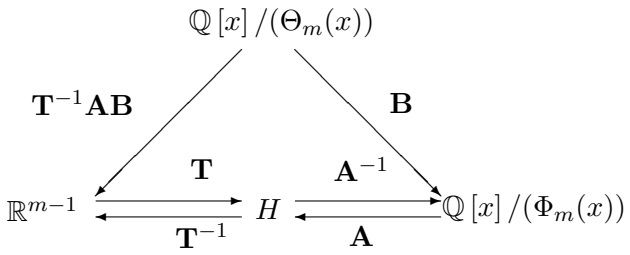

Figure 6.2: Mappings between spaces for m an odd prime.

In Figure 6.2 we give a version of Figure 6.1 where the canonical embedding into H and the isomorphism into $\mathbb{R}^{m-1}$, using the T-basis, are decoupled. All maps are given by matrices with respect to specific bases, which we now define. We regard $\mathbb{Q}[x]/(\Theta_m(x))$ as a vector space over $\mathbb{Q}$ with basis $\{1, x, \ldots, x^{m-1}\}$ and $\mathbb{Q}[x]/(\Phi_m(x))$ as a vector space over $\mathbb{Q}$ with basis $\{x, \ldots, x^{m-1}\}$. We can obviously extend these rational vector spaces to real or complex vector spaces in a natural way and use the same bases. We use the I-basis for H and the standard basis for $\mathbb{R}^{m-1}$. The mapping β from $\mathbb{Q}[x]/(\Theta_m(x))$ to $\mathbb{Q}[x]/(\Phi_m(x))$ is a straightforward modular reduction in which 1 is mapped to $-(x + \ldots x^{m-1})$. This is realised with respect to the bases given above by the $(m-1) \times m$ matrix $\mathbf{B} = (-\mathbf{1} \,|\mathbf{I}_1)$, where $\mathbf{1}$ is a column vector of 1s. Let ζ_m be a primitive m^{th} root of unity. Then the canonical embedding σ is realised by the symmetric $m_1 \times m_1$ matrix $\mathbf{A} = (A_{jk})$, where $A_{jk} = \zeta_m^{jk}$ for $1 \leq j, k \leq m-1$. The $(m-1) \times m$ matrix $\mathbf{E} = \mathbf{T}^{-1}\mathbf{AB}$ represents the map denoted $\mathbf{T}^{-1} \circ \gamma = \mathbf{T}^{-1} \circ \sigma \circ \beta$.

Claim (i) in the proof of [95, Theorem 5]. We now show that Claim (i) in the proof of [95, Theorem 5] holds in the case that m is an odd prime. That is, we prove that $\mathbf{G}\overline{\mathbf{G}^T} = m\mathbf{I}$, based on the argument as given in [95]. Recall that $\mathbf{G}$ is a matrix denoting the composition map $\gamma = \sigma \circ \beta$ so in this case we may write $\mathbf{G} = \mathbf{AB}$ as a matrix representing this map for our chosen bases. We see that $\mathbf{G} = \mathbf{A}\,(-\mathbf{1} \,|\mathbf{I}_1) = (\mathbf{1} \,|\mathbf{A})$, so we have

$$\mathbf{G}\overline{\mathbf{G}^T} = \begin{pmatrix} 1 & \zeta_m & \ldots & \zeta_m^{m-1} \\ 1 & (\zeta_m^2) & \ldots & (\zeta_m^2)^{m-1} \\ \vdots & & & \vdots \\ 1 & \overline{\zeta_m^2} & \ldots & (\overline{\zeta_m}^2)^{m-1} \\ 1 & \overline{\zeta_m} & \ldots & (\overline{\zeta_m})^{m-1} \end{pmatrix} \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \overline{\zeta_m} & \left(\overline{\zeta_m}\right)^2 & \ldots & \zeta_m \\ (\overline{\zeta_m})^2 & (\overline{\zeta_m}^2)^2 & \ldots & \zeta_m^2 \\ \vdots & \vdots & & \vdots \\ (\overline{\zeta_m})^{m-1} & (\overline{\zeta_m^2})^{m-1} & \ldots & \zeta_m^{m-1} \end{pmatrix}.$$

The i^{th} row of $\mathbf{G}$ contains the root ζ_m^i raised to all powers j for $1 \leq j \leq m-1$. This leads to a permutation of the roots, so each row contains all the m^{th} roots of unity in some order. A list of all roots is still a list of all roots after conjugation, so the columns of $\overline{\mathbf{G}^T}$ similarly contain all roots.

Consider the $(i,j)^{th}$ element of $\mathbf{G}\overline{\mathbf{G}^T}$. This is equal to

$$1 + \zeta_m^i \overline{\zeta_m}^j + (\zeta_m^i \overline{\zeta_m}^j)^2 + \cdots + (\zeta_m^i \overline{\zeta_m}^j)^{m-1} = \begin{cases} m & [i=j] \\ 0 & [i \neq j] \end{cases} .$$

When $i = j$ we have $1 + (\zeta_m\overline{\zeta_m})^i + ((\zeta_m\overline{\zeta_m})^i)^2 + \cdots + ((\zeta_m\overline{\zeta_m})^i)^{m-1}$. A root of unity multiplied by its conjugate is 1, so the $(i,i)^{th}$ element of $\mathbf{G}\overline{\mathbf{G}^T}$ is equal to $1 + \cdots + 1 = m$. When $i \neq j$ consider each summand. The value $\zeta_m^i \overline{\zeta_m}^j$ is equal to a particular m^{th} root of unity not equal to 1, and in each summand it is raised to a different power in $1, 2, \ldots m-1$. The resulting expression is a geometric progression and hence the sum is

$$\frac{\left(\zeta_m^i \overline{\zeta_m}^j\right)^m - 1}{\zeta_m^i \overline{\zeta_m}^j - 1} = 0 .$$

This shows that $\mathbf{G}\overline{\mathbf{G}^T} = m\mathbf{I}$ when m is an odd prime.

Analysis of the map given by the matrix $\mathbf{T}^{-1}\mathbf{AB}$ on spherical Gaussians. By the result of Ducas and Durmus [95, Theorem 5], the matrix $\mathbf{T}^{-1}\mathbf{AB}$ represents a real-valued map that sends a spherical Gaussian in $\mathbb{Q}[x]/(\Theta_m(x))$ of standard deviation s to a spherical Gaussian in $\mathbb{R}^{m-1}$ of standard deviation $s\sqrt{m'} = s\sqrt{m}$, since for odd m we have $m' = m$. In this section we consider the effect of the map given by $\mathbf{T}^{-1}\mathbf{AB}$ on a spherical Gaussian in more detail.

We first consider the effect of the map represented by the matrix $\mathbf{B} = (-\mathbf{1}|\mathbf{I}_1)$ on a spherical Gaussian.

Lemma 27. *Let $\mathbf{I}_1$ be the $m_1 \times m_1$ identity matrix, let $\mathbf{B} = (-\mathbf{1}|\mathbf{I}_1)$, and let $\mathbf{K}_1$ be the $m_1 \times m_1$ matrix with every entry 1. Then $\mathbf{B}\mathbf{B}^T = \mathbf{I}_1 + \mathbf{K}_1$.*

Proof: By direct computation we see that

$$\mathbf{B}\mathbf{B}^T = (-\mathbf{1}|\mathbf{I}_1)\begin{pmatrix} -\mathbf{1}^T \\ \hline I_1 \end{pmatrix} = \begin{pmatrix} 2 & & 1 \\ & \ddots & \\ 1 & & 2 \end{pmatrix} = \mathbf{I}_1 + \mathbf{K}_1 .$$

□

If $X \sim \mathrm{N}_m(0, \rho^2 \mathbf{I}_0)$ is spherically symmetric zero-mean real-valued m-dimensional normal random variable with component variance ρ^2, then mapping this under β, we see $\mathbf{B}X \sim \mathrm{N}_{m_1}(0, \rho^2 \mathbf{B}\mathbf{B}^T)$. By diagonalising $\mathbf{B}\mathbf{B}^T$, which has eigenvalues m (with multiplicity 1) and 1 (with multiplicity $m-2$), we see that a spherically symmetric error distribution in $\mathbb{Q}[x]/(\Theta_m(x))$ gives an error distribution in $\mathbb{Q}[x]/(\Phi_m(x))$ that is highly elliptical.

We conclude with an alternative proof of [95, Theorem 5] in the case that m is an odd prime. In order to do so, we require the following lemmas.

Lemma 28. *Let $\mathbf{A}$ be the symmetric $m_1 \times m_1$ matrix $\mathbf{A} = (A_{jk})$, where $A_{jk} = \zeta_m^{jk}$ for $1 \leq j, k \leq m-1$, let $\mathbf{I}_1$ be the $m_1 \times m_1$ identity matrix and let $\mathbf{K}_1$ be the $m_1 \times m_1$ matrix with every entry 1. Then $\mathbf{A}\overline{\mathbf{A}} = m\mathbf{I}_1 - \mathbf{K}_1$.*

Proof: Considering each entry of $\mathbf{A}\overline{\mathbf{A}}$, we see that

$$\left(\mathbf{A}\overline{\mathbf{A}}\right)_{jk} = \sum_{l=1}^{m-1} A_{jl}\overline{A_{lk}} = \sum_{l=1}^{m-1} \zeta_m^{jl}\zeta_m^{-lk} = \sum_{l=1}^{m-1} \zeta_m^{l(j-k)} = \begin{cases} m-1 & [j=k] \\ -1 & [j \neq k]. \end{cases}$$

□

Lemma 29. *Let $\mathbf{A}$ be the symmetric $m_1 \times m_1$ matrix $\mathbf{A} = (A_{jk})$, where $A_{jk} = \zeta_m^{jk}$ for $1 \leq j, k \leq m-1$, let $\mathbf{J}_1$ be the $m_1 \times m_1$ matrix with 1s on the reverse diagonal and 0 elsewhere, and let $\mathbf{K}_1$ be the $m_1 \times m_1$ matrix with every entry 1. Then $\mathbf{A}\mathbf{A} = m\mathbf{J}_1 - \mathbf{K}_1$.*

Proof: Considering each entry of $\mathbf{A}\mathbf{A}$, we see that

$$\left(\mathbf{A}\mathbf{A}\right)_{jk} = \sum_{l=1}^{m-1} A_{jl}A_{lk} = \sum_{l=1}^{m-1} \zeta_m^{jl}\zeta_m^{lk} = \sum_{l=1}^{m-1} \zeta_m^{l(j+k)} = \begin{cases} m-1 & [j=m-k] \\ -1 & [j \neq m-k]. \end{cases}$$

□

Lemma 30. *Let $\mathbf{A}$ be the symmetric $m_1 \times m_1$ matrix $\mathbf{A} = (A_{jk})$, where $A_{jk} = \zeta_m^{jk}$ for $1 \leq j, k \leq m-1$, and let $\mathbf{K}_1$ be the $m_1 \times m_1$ matrix with every entry 1. Then $\mathbf{A}\mathbf{K}_1 = -\mathbf{K}_1$.*

Proof: Considering each entry of $\mathbf{AK}_1$, we see that

$$(\mathbf{AK}_1)_{jk} = \sum_{l=1}^{m-1} A_{jl}(K_1)_{lk} = \sum_{l=1}^{m-1} A_{jl} \cdot 1 = \sum_{l=1}^{m-1} \zeta_m^{jl} = -1\,.$$

□

Lemma 31. *Let* $\mathbf{A}$ *be the symmetric* $m_1 \times m_1$ *matrix* $\mathbf{A} = (A_{jk})$*, where* $A_{jk} = \zeta_m^{jk}$ *for* $1 \leq j, k \leq m-1$*, and let* $\mathbf{K}_1$ *be the* $m_1 \times m_1$ *matrix with every entry* 1*. Then* $-\mathbf{K}_1\mathbf{A} = \mathbf{K}_1$*.*

Proof: Using Lemma 30, and the fact that $\mathbf{A}$ and $\mathbf{K}_1$ are symmetric, we obtain

$$\begin{aligned} \mathbf{AK}_1 &= -\mathbf{K}_1 \\ (\mathbf{AK}_1)^T &= -\mathbf{K}_1^T \\ \mathbf{K}_1\mathbf{A} &= -\mathbf{K}_1\,, \end{aligned}$$

from which the result follows. □

Lemma 32. *Let* $\mathbf{A}$ *be the symmetric* $m_1 \times m_1$ *matrix* $\mathbf{A} = (A_{jk})$*, where* $A_{jk} = \zeta_m^{jk}$ *for* $1 \leq j, k \leq m-1$*, and let* $\mathbf{K}_1$ *be the* $m_1 \times m_1$ *matrix with every entry* 1*. Then* $-\mathbf{K}_1\overline{\mathbf{A}} = \mathbf{K}_1$*.*

Proof: Using Lemma 30, the fact that $\mathbf{K}_1$ is real, and the fact that $\overline{\mathbf{A}}$ and $\mathbf{K}_1$ are symmetric, we obtain

$$\begin{aligned} \mathbf{AK}_1 &= -\mathbf{K}_1 \\ \overline{\mathbf{AK}_1} &= -\overline{\mathbf{K}_1} \\ (\overline{\mathbf{A}} \cdot \mathbf{K}_1)^T &= (-\mathbf{K}_1)^T\,, \end{aligned}$$

from which the result follows. □

Lemma 33. *Let* $\mathbf{J}_1$ *be the* $m_1 \times m_1$ *matrix with* 1*s on the reverse diagonal and* 0 *elsewhere and let* $\mathbf{T}$ *be the* $m_1 \times m_1$ *matrix of Definition 26. Then* $\mathbf{J}_1\overline{\mathbf{T}} = \mathbf{T}$*.*

Proof: The j^{th} row of $\mathbf{J}_1$ has entries $(J_1)_{j,l} = 1$ for $l = m-j$ and $(J_1)_{j,l} = 0$ otherwise. So, when multiplying any $m_1 \times m_1$ matrix M on the left by $\mathbf{J}_1$, the entries in the product are given by:

$$(\mathbf{J}_1\mathbf{M})_{jk} = \sum_{l=1}^{m-1} (J_1)_{jl} M_{lk} = M_{m-j,k}\,.$$

So $\mathbf{J}_1$ reflects the rows of $\mathbf{M}$ about a line at the midpoint of the matrix.

Let $\mathbf{I}_2$ and $\mathbf{J}_2$ be the $m_2 \times m_2$ versions of $\mathbf{I}_1$ and $\mathbf{J}_1$ respectively. By definition $\mathbf{T} = \frac{1}{\sqrt{2}} \left(\begin{array}{c|c} \mathbf{I}_2 & i\mathbf{J}_2 \\ \hline \mathbf{J}_2 & -i\mathbf{I}_2 \end{array} \right)$ and its inverse is $\mathbf{T}^{-1} = \frac{1}{\sqrt{2}} \left(\begin{array}{c|c} \mathbf{I}_2 & \mathbf{J}_2 \\ \hline -i\mathbf{J}_2 & i\mathbf{I}_2 \end{array} \right)$. By Claim 2 in the proof of [95, Theorem 5],

$$\overline{\mathbf{T}} = (\mathbf{T}^{-1})^T = \frac{1}{\sqrt{2}} \left(\begin{array}{c|c} \mathbf{I}_2 & -i\mathbf{J}_2 \\ \hline \mathbf{J}_2 & i\mathbf{I}_2 \end{array} \right) .$$

So the reflection $\mathbf{J}_1\overline{\mathbf{T}}$ is given by

$$\mathbf{J}_1\overline{\mathbf{T}} = \frac{1}{\sqrt{2}} \left(\begin{array}{c|c} \mathbf{I}_2 & i\mathbf{J}_2 \\ \hline \mathbf{J}_2 & -i\mathbf{I}_2 \end{array} \right) = \mathbf{T} .$$

□

We can now restate and give another proof of the result of Ducas and Durmus [95, Theorem 5] in the case that m is an odd prime.

Theorem 14 ([95])**.** *Let* $\mathbf{T}$*,* $\mathbf{A}$ *and* $\mathbf{B}$ *be as defined above. Then the matrix* $\mathbf{T}^{-1}\mathbf{AB}$ *represents a real-valued map sending a spherical Gaussian of standard deviation* s *in* $\mathbb{Q}[x]/(\Theta_m(x))$ *to a spherical Gaussian of standard deviation* $s\sqrt{m'} = s\sqrt{m}$ *in* $\mathbb{R}^{m-1}$*.*

Proof: Recall that $\mathbf{T}^{-1}$ maps H considered as a subset of $\mathbb{C}^{m-1}$ to $\mathbb{R}^{m-1}$ considered as a subset of $\mathbb{C}^{m-1}$. Consider $\mathbb{Q}[x]/(\Theta_m(x))$ also as a subset of a complex space. Let Z be a spherically symmetric real-valued m-dimensional normal random variable with component variance s^2 and mean 0, and let $\mathbf{I}_0$ denote the $m \times m$ identity matrix. Then Z can be expressed as a complex normal random variable by $Z \sim \mathbb{C}\mathrm{N}_m(0, s^2\mathbf{I}_0, s^2\mathbf{I}_0)$, where both the covariance matrix $\mathrm{E}\left[Z\overline{Z}^T\right]$ and the relation matrix $\mathrm{E}\left[ZZ^T\right]$ are $s^2\mathbf{I}_0$.

The complex normal random variable $Y = \mathbf{T}^{-1}\mathbf{A}\mathbf{B}Z$ has covariance matrix

$$\begin{aligned}
\mathrm{E}\left[Y\overline{Y}^T\right] = s^2\mathbf{T}^{-1}\mathbf{A}\mathbf{B}\left(\overline{\mathbf{T}^{-1}\mathbf{A}\mathbf{B}}\right)^T &= s^2\mathbf{T}^{-1}\mathbf{A}(\mathbf{B}\mathbf{B}^T)\overline{\mathbf{A}}\left(\overline{\mathbf{T}^{-1}}\right)^T \\
&= s^2\mathbf{T}^{-1}\mathbf{A}(\mathbf{I}_1 + \mathbf{K}_1)\overline{\mathbf{A}}\mathbf{T} \\
&= s^2\mathbf{T}^{-1}(\mathbf{A}\mathbf{I}_1\overline{\mathbf{A}} + \mathbf{A}\mathbf{K}_1\overline{\mathbf{A}})\mathbf{T} \\
&= s^2\mathbf{T}^{-1}(\mathbf{A}\overline{\mathbf{A}} + \mathbf{A}\mathbf{K}_1\overline{\mathbf{A}})\mathbf{T} \\
&= s^2\mathbf{T}^{-1}(m\mathbf{I}_1 - \mathbf{K}_1 + (\mathbf{A}\mathbf{K}_1)\overline{\mathbf{A}})\mathbf{T} \\
&= s^2\mathbf{T}^{-1}(m\mathbf{I}_1 - \mathbf{K}_1 - \mathbf{K}_1\overline{\mathbf{A}})\mathbf{T} \\
&= s^2\mathbf{T}^{-1}(m\mathbf{I}_1 - \mathbf{K}_1 + \mathbf{K}_1)\mathbf{T} \\
&= s^2\mathbf{T}^{-1}(m\mathbf{I}_1)\mathbf{T}\,,
\end{aligned}$$

where to obtain the second line we used Lemma 27, to obtain the fifth line we used Lemma 28, to obtain the sixth line we used Lemma 30 and to obtain the seventh line we used Lemma 32.

The complex normal random variable $Y = \mathbf{T}^{-1}\mathbf{A}\mathbf{B}Z$ has relation matrix

$$\begin{aligned}
\mathrm{E}\left[YY^T\right] = s^2\mathbf{T}^{-1}\mathbf{A}\mathbf{B}\left(\mathbf{T}^{-1}\mathbf{A}\mathbf{B}\right)^T &= s^2\mathbf{T}^{-1}\mathbf{A}(\mathbf{B}\mathbf{B}^T)\mathbf{A}\left(\mathbf{T}^{-1}\right)^T \\
&= s^2\mathbf{T}^{-1}\mathbf{A}(\mathbf{I}_1 + \mathbf{K}_1)\mathbf{A}\overline{\mathbf{T}} \\
&= s^2\mathbf{T}^{-1}(\mathbf{A}\mathbf{I}_1\mathbf{A} + \mathbf{A}\mathbf{K}_1\mathbf{A})\overline{\mathbf{T}} \\
&= s^2\mathbf{T}^{-1}(\mathbf{A}\mathbf{A} + \mathbf{A}\mathbf{K}_1\mathbf{A})\overline{\mathbf{T}} \\
&= s^2\mathbf{T}^{-1}(m\mathbf{J}_1 - \mathbf{K}_1 + (\mathbf{A}\mathbf{K}_1)\mathbf{A})\overline{\mathbf{T}} \\
&= s^2\mathbf{T}^{-1}(m\mathbf{J}_1 - \mathbf{K}_1 + (-\mathbf{K}_1\mathbf{A}))\overline{\mathbf{T}} \\
&= s^2\mathbf{T}^{-1}(m\mathbf{J}_1 - \mathbf{K}_1 + \mathbf{K}_1)\overline{\mathbf{T}} \\
&= s^2\mathbf{T}^{-1}m(\mathbf{J}_1\overline{\mathbf{T}}) \\
&= s^2\mathbf{T}^{-1}m\mathbf{T}\,,
\end{aligned}$$

where we again used Lemma 27 to obtain the second line and Lemma 30 for the sixth line; and we used Lemma 29 to obtain the fifth line, Lemma 31 to obtain the seventh line, and Lemma 33 to obtain the ninth line.

Thus we have $Y = \mathbf{T}^{-1}\mathbf{A}\mathbf{B}Z \sim \mathbb{C}\mathrm{N}_{m_1}(0, ms^2\mathbf{I}_1, ms^2\mathbf{I}_1)$ is a complex normal random variable with identical covariance matrix and relation matrix which implies it is real-valued. In particular it is a zero-mean spherically symmetric real-valued random variable with component variance ms^2, from which the result follows. □

Chapter 7

Homomorphic Encryption

Contents

This chapter considers Ring-LWE-based Fully Homomorphic Encryption schemes. We consider the choice of parameters for both security and correctness in this setting. We focus on SEAL [93, 146, 58, 59, 60], a somewhat homomorphic encryption library developed by Microsoft Research that implements the FV scheme of Fan and Vercauteren [100]. This chapter is based on material presented in [146, 58, 59, 60].

7.1 Introduction to FHE

Several encryption schemes have the property that we can perform certain operations on ciphertexts, without having access to the secret key, and have these operations

translate meaningfully to operations on the underlying plaintexts. For example, when we multiply two ElGamal [98] ciphertexts, we obtain a ciphertext encrypting the product of the underlying plaintexts. As a second example, when we multiply two Paillier [184] ciphertexts, we obtain a ciphertext encrypting the sum of the underlying plaintexts.

Suppose an encryption scheme could be augmented with a *homomorphic addition* operation that when applied to two input ciphertexts would produce an output ciphertext encrypting the sum of the underlying plaintexts of the input ciphertexts. Suppose this same scheme could also be augmented with a *homomorphic multiplication* operation that when applied to two input ciphertexts would produce an output ciphertext encrypting the product of the underlying plaintexts of the input ciphertexts. This scheme would then enable arbitrary computation on encrypted data. Constructing such a *fully homomorphic encryption* scheme was a longstanding open problem since its proposal by Rivest *et al.* [198] in 1978. It was resolved in 2009 in Gentry's seminal work [107].

Homomorphic encryption enables a range of applications [174], which are typically described in the client-server model. Homomorphic encryption allows a computationally weak client to outsource computation on their data to a powerful untrusted server. The client encrypts their data and passes it to the server, along with a description of the computation they wish to perform. The server operates on the data homomorphically without access to the secret key, and produces an output ciphertext that is sent back to the client. The client decrypts this ciphertext to determine the result of the computation. Since the server only sees an encryption of the data and never has access to the secret key, the client can be assured that the server does not learn anything about their data, or indeed the output of the computation.

Gentry's original scheme [107] begins by specifying a *somewhat homomorphic encryption* scheme, in which ciphertexts have a *noise*, which grows during homomorphic operations. Gentry then shows how to transform a somewhat homomorphic encryption scheme into a fully homomorphic encryption scheme using a technique known as *bootstrapping*. A large number of somewhat homomorphic cryptosystems have since been proposed in the literature, for example [41, 42, 44, 48, 47, 100, 112, 153, 212]. Several of these [42, 44, 47, 100] are based on Ring-LWE.

In the remainder of this thesis we turn our attention to parameter selection in Ring-LWE-based homomorphic encryption. We focus on the FV scheme of Fan and Vercauteren [100], as implemented in SEAL [93] since version 2.0 [146, 58, 60]. We describe this scheme in Section 7.1.1.

Ring-LWE-based FHE is particularly interesting from the perspective of selecting parameters for security as certain attacks may be more effective in this setting. For example, Albrecht [7] showed his improvements to the standard distinguishing attack are very effective on the homomorphic encryption implementations HElib [121] and SEAL v2.0 [146]. We discuss security of Ring-LWE-based FHE schemes in Section 7.2.

A key issue in Ring-LWE based FHE is appropriate encoding (and corresponding decoding) of raw data into the plaintext space, which is typically a polynomial ring. The choice of plaintext space and encoder can affect the performance of the scheme dramatically. We discuss encoding in detail in Section 7.3.

In homomorphic encryption the setting of parameters to ensure correctness is crucial, due to the presence of noise in ciphertexts. The noise is typically small in a fresh ciphertext, and grows as homomorphic evaluation operations are performed. If the noise grows too large, then decryption fails, so a good understanding of the noise growth behaviour of a homomorphic encryption scheme is essential to choose appropriate parameters to ensure correctness. In schemes based on LWE and Ring-LWE, this typically means choosing a very large q coupled with very narrow error distributions, which is unusual and is another reason why studying the security is so interesting. We study the noise growth behaviour in SEAL in Section 7.4 respectively.

We conclude by focussing in Sections 7.5 and 7.6 on parameter selection in SEAL for security, performance and correctness.

7.1.1 SEAL: Implementing the FV scheme

In this section we introduce SEAL, a homomorphic encryption library developed by Microsoft Research. In application settings such as bioinformatics, the potential user

of homomorphic encryption is not necessarily an expert in cryptography. SEAL aims to be accessible to this audience by providing a well-engineered and well-documented homomorphic encryption library that would be equally easy to use both by experts in cryptography and by those with little or no cryptographic background. SEAL was first described by Dowlin *et al.* [93] and the current version at the time of writing is SEAL v2.2 [60]. Since SEAL v2.0 [146] the underlying encryption scheme has been the FV scheme of Fan and Vercauteren [100]. Previously the underlying encryption scheme was YASHE, which was proposed by Bos *et al.* [41].

A number of other homomorphic encryption libraries exist, such as HElib [121, 122, 123], PALISADE [2], cuHE [85], TFHE [73] and HEAAN [69]. Homomorphic encryption is implemented as an example application in the lattice library $\Lambda \circ \lambda$ [83]. Among these homomorphic encryption libraries, PALISADE is the only one that also implements the FV scheme.

Other implementations of FV can be found. For example, an implementation of FV in R is given by Aslett *et al.* [26]. FV-NFLlib [1] implements FV on top of the ideal lattice library NFLlib [163]. Bajard *et al.* [31] propose a variant of FV with adaptations to the homomorphic multiplication and decryption operations to enable a Chinese Remainder Theorem representation of ciphertexts. They provide an implementation, which is also based on NFLlib, and show that their variant is faster than FV-NFLlib.

Among the most comparable projects to SEAL is the C++ library HElib, which implements the BGV scheme [44]. The FV scheme and the BGV scheme are both based on variants of Ring-LWE. Comparing the respective implementations of BGV as in HElib and of FV as in SEAL would be a very interesting research direction, but appears challenging. Many of the challenges relate to appropriate parameter selection: on the security side, we want to ensure the underlying Ring-LWE instance is similar; and at the same time we want to pick optimal parameters for performance. We also want to ensure that we are able to correctly decrypt the output of the homomorphic evaluation operation being computed.

Table 7.1 lists the parameters that the user in SEAL can select, as well as other related parameters. For many parameters, default choices are provided for which

Parameter	Description	Name in SEAL (if applicable)
q	Modulus in the ciphertext space (coefficient modulus)	`coeff_modulus`
t	Modulus in the plaintext space (plaintext modulus)	`plain_modulus`
n	A power of 2	
$x^n + 1$	The polynomial modulus that specifies the ring R	`poly_modulus`
R	The ring $\mathbb{Z}[x]/(x^n + 1)$	
R_a	The ring $\mathbb{Z}_a[x]/(x^n + 1)$, that is, the ring R but with coefficients reduced modulo a	
w	A base into which ciphertext elements are decomposed during relinearization	
$\log w$		`decomposition_bit_count`
ℓ	There are $\ell + 1 = \lfloor \log_w q \rfloor + 1$ elements in each component of each evaluation key	
δ	Expansion factor in the ring R ($\delta \leq n$)	
Δ	Quotient on division of q by t, that is, $\lfloor q/t \rfloor$	
$r_t(q)$	Remainder on division of q by t, that is, $q = \Delta t + r_t(q)$ for $0 \leq r_t(q) < t$	
χ	Error distribution (a truncated discrete Gaussian distribution)	
σ	Standard deviation of χ	`noise_standard_deviation`
B	Bound on the distribution χ	`noise_max_deviation`

Table 7.1: Parameters in SEAL.

performance is expected to be good. We discuss this in detail in Section 7.5. On the other hand, the documentation available for HElib [121, 122, 123] does not make clear how to select optimal parameters for performance, so it is difficult to assure that a performance comparison with SEAL is fair.

A related issue is that the parameters that the user is required to select in SEAL, and those that have a default value, are different to those in HElib. It is unclear whether it is more reasonable to keep the default choices, which may have been chosen for good performance; or to alter them, for example to choose parameters that imply similar underlying Ring-LWE instances. For example, the default secret key distribution in HElib is sparse by default [121], whereas this is not the case in SEAL. This sparseness potentially enables more attacks, so could be seen as worse for security. At the same time, the sparseness reduces noise growth and hence may enable more homomorphic evaluation operations to be performed, so could be seen as better for performance. We discuss the noise growth behaviour in SEAL in detail in Section 7.4.

Costache and Smart [76] give a theoretical comparison of the FV, BGV, and YASHE schemes as well as the LTV scheme [153, 91]. Several works compare two or more different homomorphic encryption schemes for their suitability for a specific task. For example, Lepoint and Naehrig [149] compare the use of FV and YASHE for the homomorphic evaluation of SIMON [32]. Kim and Lauter [138] compare the use of BGV and YASHE for computations on genomic data.

To the best of our knowledge, there are no works that compare two or more different implementations of the same homomorphic encryption scheme, for a specific task or otherwise. Such a comparison would be an interesting direction for future work. We would expect that, for example, a comparison between SEAL and FV-NFLlib would be challenging for similar reasons as discussed above for a comparison between SEAL and HElib, such as ensuring that the choice of parameters gives optimal performance for both implementations.

7.1.1.1 The FV Scheme

We now give the definition of the FV scheme as originally presented by Fan and Vercauteren [100] and in the following subsection we describe how SEAL implements a slightly more general version of FV. The parameters n, t, q, w and ℓ are as described in Table 7.1.

In FV the plaintext space is $R_t = \mathbb{Z}_t[x]/(x^n + 1)$, that is, polynomials of degree less than n with coefficients modulo t. Ciphertexts in FV are pairs of polynomials in $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. We denote an FV ciphertext $\mathtt{ct} = (\mathtt{ct}[0], \mathtt{ct}[1])$. In FV, the secret key is sampled from the space $R_2 = \{0, 1\}[x]/(x^n + 1)$.

Let λ be the security parameter. Let w be a base, and let $\ell+1 = \lfloor \log_w q \rfloor + 1$ denote the number of terms in the decomposition into base w of an integer in base q. We will decompose polynomials in R_q into base-w components coefficient-wise, resulting in $\ell + 1$ polynomials.

The FV scheme contains the following algorithms, described below: `SecretKeyGen`, `PublicKeyGen`, `EvaluationKeyGen`, `Encrypt`, `Decrypt`, `Add`, and `Multiply`.

- `SecretKeyGen`(λ): Sample $s \xleftarrow{\$} R_2$ and output
$$\mathtt{sk} = s\,.$$
- `PublicKeyGen`(`sk`): Set $s = \mathtt{sk}$, sample $a \xleftarrow{\$} R_q$ and $e \leftarrow \chi$. Output
$$\mathtt{pk} = ([-(as + e)]_q, a)\ .$$
- `EvaluationKeyGen`(`sk`, w): for $i \in \{0, \ldots, \ell\}$, sample $a_i \xleftarrow{\$} R_q$ and $e_i \leftarrow \chi$. Output
$$\mathtt{evk} = \left([-(a_i s + e_i) + w^i s^2]_q, a_i\right)\ .$$
- `Encrypt`(`pk`, m): For the message $m \in R_t$. Let $\mathtt{pk} = (p_0, p_1)$, sample $u \xleftarrow{\$} R_2$ and $e_1, e_2 \leftarrow \chi$. Output
$$\mathtt{ct} = ([\Delta m + p_0 u + e_1]_q, [p_1 u + e_2]_q)\ .$$

- $\texttt{Decrypt}(\texttt{sk}, \texttt{ct})$: Set $s = \texttt{sk}$. Let $\texttt{ct}[0] = c_0$ and $\texttt{ct}[1] = c_1$. Output

$$\left[\left\lfloor \frac{t}{q}[c_0 + c_1 s]_q \right\rceil\right]_t .$$

- $\texttt{Add}(\texttt{ct}_0, \texttt{ct}_1)$: Output

$$(\texttt{ct}_0[0] + \texttt{ct}_1[0], \texttt{ct}_0[1] + \texttt{ct}_1[1]) .$$

- $\texttt{Multiply}(\texttt{ct}_0, \texttt{ct}_1)$: Compute

$$\begin{aligned} c_0 &= \left[\left\lfloor \frac{t}{q}\texttt{ct}_0[0]\texttt{ct}_1[0] \right\rceil\right]_q , \\ c_1 &= \left[\left\lfloor \frac{t}{q}\left(\texttt{ct}_0[0]\texttt{ct}_1[1] + \texttt{ct}_0[1]\texttt{ct}_1[0]\right) \right\rceil\right]_q , \\ c_2 &= \left[\left\lfloor \frac{t}{q}\texttt{ct}_0[1]\texttt{ct}_1[1] \right\rceil\right]_q . \end{aligned}$$

  Let $\texttt{evk}[i][0] = [-(a_i s + e_i) + w^i s^2]_q$ and $\texttt{evk}[i][1] = a_i$. Express c_2 in base w as $c_2 = \sum_{i=0}^{\ell} c_2^{(i)} w^i$. Set

$$\begin{aligned} c_0' &= c_0 + \sum_{i=0}^{\ell} \texttt{evk}[i][0] c_2^{(i)}, \\ c_1' &= c_1 + \sum_{i=0}^{\ell} \texttt{evk}[i][1] c_2^{(i)}. \end{aligned}$$

  Output

$$(c_0', c_1') .$$

7.1.1.2 How SEAL differs from FV

Some operations in SEAL are done slightly differently, or in slightly more generality, than in textbook FV. In this section we discuss these differences in detail.

Plaintext space and ciphertext space. Plaintext elements in SEAL are polynomials in R_t, just as in textbook FV. Ciphertexts in SEAL are formed of polynomials in R_q. In FV ciphertexts are formed of a pair of polynomials, so we say the ciphertext has size 2. In SEAL a ciphertext of any size greater than or equal 2 can be valid. Note that because we consider well-formed ciphertexts of arbitrary length

valid, we automatically lose the *compactness* property of homomorphic encryption. Roughly speaking, compactness states that the decryption circuit should not depend on ciphertexts, or on the function being evaluated [24].

Key Distribution. In textbook FV the secret key is a polynomial sampled uniformly from R_2 so it is a polynomial with coefficients in $\{0, 1\}$. In SEAL the secret key is sampled uniformly from R_3 so it is a polynomial with coefficients in $\{-1, 0, 1\}$.

Decryption. A SEAL ciphertext $\texttt{ct} = (c_0, \ldots, c_k)$ is decrypted by computing

$$\left[\left\lfloor \frac{t}{q}[\texttt{ct}(s)]_q \right\rceil\right]_t = \left[\left\lfloor \frac{t}{q}\left[c_0 + \cdots + c_k s^k\right]_q \right\rceil\right]_t .$$

In textbook FV, decryption of a ciphertext $\texttt{ct} = (c_0, c_1)$ involves computing $c_0 + c_1 s$. To make clear the generalisation of FV in SEAL it is convenient to think of each ciphertext component as corresponding to a particular power of the secret key s, so that the c_i term is associated with s^i for $0 \leq i \leq k$, as can be seen in the decryption description. This generalises the situation in textbook FV, where the c_0 term is associated with $s^0 = 1$ and the c_1 term is associated with s^1.

Multiplication. The `Multiply` function in FV has two steps: the first step computes the intermediate object (c_0, c_1, c_2), and the second step uses the evaluation key to transform this into an output ciphertext (c'_0, c'_1). Such a transformation is called *relinearization*. In fact, the intermediate object (c_0, c_1, c_2) can be decrypted using the generalised decryption formula above. Hence, the relinearization step is not necessary for correctness of homomorphic multiplication and (c_0, c_1, c_2) is a valid ciphertext in SEAL. Nonetheless, (c_0, c_1, c_2) is not a valid ciphertext in FV because it is of size 3, and so relinearization is necessary in FV to obtain a valid ciphertext (c'_0, c'_1) of size 2.

In SEAL, the two steps are decoupled. The `Multiply` function implements the first step and outputs the intermediate ciphertext. The function `Relinearize` implements the second step and outputs a ciphertext of smaller size encrypting the same underlying plaintext.

Indeed SEAL allows a generalised version of the first step of multiplication so even larger ciphertexts can be produced. In particular, let $\mathtt{ct}_1 = (c_0, c_1, \ldots, c_j)$ and $\mathtt{ct}_2 = (d_0, d_1, \ldots, d_k)$ be two SEAL ciphertexts of sizes $j+1$ and $k+1$, respectively. Let the ciphertext $\mathtt{ct}_{\text{mult}}$ output by $\mathtt{Multiply}(\mathtt{ct}_1, \mathtt{ct}_2)$, which is of size $j+k+1$, be denoted $\mathtt{ct}_{\text{mult}} = (C_0, C_1, \ldots, C_{j+k})$. The polynomials $C_m \in R_q$ are computed as

$$C_m = \left[\left\lfloor \frac{t}{q} \left(\sum_{r+s=m} c_r d_s \right) \right\rceil \right]_q .$$

Relinearization. The goal of relinearization in FV is to decrease the size of the ciphertext back to 2 after it has been increased by multiplication. Since ciphertexts in SEAL are allowed to grow arbitrarily in size, and it is possible to decrypt a ciphertext of any size at least 2, the relinearization operation is generalised accordingly. In particular, the function `Relinearize` in SEAL takes an input ciphertext of any size k, and appropriate evaluation keys, and outputs a ciphertext of size j where $2 \leq j < k$. The output ciphertext encrypts the same message as the input ciphertext, but can be decrypted using a smaller degree decryption function.

We give an intuition of the relinearization process in FV (the second step of the multiplication operation) and explain how it is generalised in SEAL. Suppose we have a size 3 ciphertext (c_0, c_1, c_2) that we want to convert into a size 2 ciphertext (c_0', c_1') that decrypts to the same result. Suppose we are also given a candidate evaluation key pair $\mathtt{evk} = \big([-(as+e)+s^2]_q, a\big)$, where $a \xleftarrow{\$} R_q$, and $e \leftarrow \chi$. Now set $c_0' = c_0 + \mathtt{evk}[0]c_2$, $c_1' = c_1 + \mathtt{evk}[1]c_2$, and define the output to be the pair (c_0', c_1'). Interpreting this as a size 2 ciphertext and decrypting it yields

$$c_0' + c_1' s = c_0 + (-(as+e)+s^2)c_2 + c_1 s + a c_2 s = c_0 + c_1 s + c_2 s^2 - e c_2 .$$

This is almost what is needed, that is to say $c_0 + c_1 s + c_2 s^2$, except for the extra additive term ec_2. This can be considered as the noise incurred in relinearization: as with any homomorphic evaluation operation, the relinearization process adds noise. Unfortunately, since c_2 has coefficients up to size q, the noise added may be extremely large and thus we would expect decryption to fail.

Instead we use the classical solution of writing c_2 in terms of some smaller base w (see for example [48, 44, 42, 100]) as $c_2 = \sum_{i=0}^{\ell} c_2^{(i)} w^i$. Instead of having just

one evaluation key pair as above, suppose we have $\ell + 1$ such pairs constructed as in Section 7.1.1.1. Then we can show that setting c'_0 and c'_1 as in Section 7.1.1.1 successfully replaces the large additive term that appeared in the naive approach above with a term of size linear in w.

This same idea can be generalised to relinearizing a ciphertext of any size $k + 1$ to size $k \geq 2$, as long as a generalised set of evaluation keys is generated in the $\texttt{EvaluationKeyGen}(\texttt{sk}, w)$ function. Namely, suppose we have a set of evaluation keys $\texttt{evk}_2$ (corresponding to s^2), $\texttt{evk}_3$ (corresponding to s^3) and so on up to $\texttt{evk}_k$ (corresponding to s^k), each generated as in Section 7.1.1.1. Then relinearization reduces the ciphertext $(c_0, c_1, \ldots, c_k)$ in size by one, producing $(c'_0, c'_1, \ldots, c'_{k-1})$, where

$$c'_0 = c_0 + \sum_{i=0}^{\ell} \texttt{evk}_k[i][0] c_k^{(i)} ,$$

$$c'_1 = c_1 + \sum_{i=0}^{\ell} \texttt{evk}_k[i][1] c_k^{(i)} ,$$

and $c'_j = c_j$ for $2 \leq j \leq k-1$. This process is then iterated as many times as desired.

Addition. We also need to generalise addition to be able to operate on ciphertexts of any size and this is implemented with the function `Add`. Suppose we have two SEAL ciphertexts $\texttt{ct}_1 = (c_0, \ldots, c_j)$ and $\texttt{ct}_2 = (d_0, \ldots d_k)$, encrypting plaintexts m_1 and m_2, respectively. Suppose without loss of generality that $j \leq k$. Then

$$\texttt{ct}_{\text{add}} = ([c_0 + d_0]_q, \ldots, [c_j + d_j]_q, d_{j+1}, \ldots, d_k)$$

encrypts $[m_1 + m_2]_t$.

Other operations. SEAL implements various other homomorphic operations for the user's convenience. The function `Sub` implements subtraction, which is entirely analogous to addition. Suppose $\texttt{ct}_1 = (c_0, \ldots, c_j)$ and $\texttt{ct}_2 = (d_0, \ldots d_k)$ encrypt plaintexts m_1 and m_2, respectively. Suppose without loss of generality that $j \geq k$. Then

$$\texttt{ct}_{\text{add}} = ([c_0 - d_0]_q, \ldots, [c_k - d_k]_q, c_{j+1}, \ldots, c_k)$$

encrypts $[m_1 - m_2]_t$.

There is also a `Negate` function that takes as input a ciphertext $\texttt{ct} = (c_0, c_1, \ldots, c_k)$ encrypting a message m and outputs a ciphertext $\texttt{ct}_{neg}$ encrypting $[-m]_t$. This is obtained by simply negating each component c_i for $0 \leq i \leq k$, so

$$\texttt{ct}_{neg} = (-c_0, -c_1, \ldots, -c_k)\,.$$

Some applications of homomorphic encryption involve working with data that does not need to be protected. For example, when wishing to homomorphically compute a weighted average of some encrypted data, the weights themselves may be public values. The functions `AddPlain` and `MultiplyPlain` give the user flexibility in such scenarios as they can be used to improve performance over first encrypting the plaintext and then performing the usual homomorphic addition or multiplication. For example, a `MultiplyPlain` operation incurs much less noise to the ciphertext than the usual `Multiply`.

Given as input a ciphertext $\texttt{ct} = (c_0, c_1, \ldots, c_k)$ encrypting a plaintext polynomial m and an unencrypted plaintext polynomial m_{addp}, the function `AddPlain` outputs a ciphertext $\texttt{ct}_{addp}$ encrypting $[m+m_{addp}]_t$. To form this ciphertext, we multiply m_{addp} by Δ and add this to the c_0 part of $\texttt{ct}$. That is, $\texttt{ct}_{addp} = (c_0 + \Delta m_{addp}, c_1, \ldots, c_k)$.

Similarly the function `MultiplyPlain` takes as input a ciphertext $\texttt{ct} = (c_0, c_1, \ldots, c_k)$ encrypting a plaintext polynomial m and an unencrypted plaintext polynomial m_{multp} and outputs a ciphertext $\texttt{ct}_{multp}$ encrypting $[m \cdot m_{multp}]_t$. This is obtained by multiplying each component c_i in the ciphertext by m_{multp} for $0 \leq i \leq k$; that is, $\texttt{ct}_{multp} = (m_{multp} \cdot c_0, m_{multp} \cdot c_1, \ldots, m_{multp} \cdot c_k)$.

7.2 Security of Ring-LWE based FHE schemes

The *d-sample Decision Ring-LWE problem* asks an adversary to distinguish whether a set of d samples was chosen according to a Ring-LWE distribution or uniformly at random. This reflects the situation in practice where an attacker would not necessarily have unlimited access to an oracle generating samples. It has been shown that solving the d-sample Decision Ring-LWE problem is equally as hard as solving the $(d-1)$-sample Decision Ring-LWE problem with the secret s instead sampled

from the error distribution [159]. Furthermore, in the case of LWE (not Ring-LWE), it is possible to argue [114] that the problem is equally as hard even if s is sampled from almost any narrow distribution with enough entropy, such as the uniform distribution on R_2 or R_3.

It can be shown that the FV scheme is *IND-CPA secure* [115] if Decision Ring-LWE with binary secret is hard. For brevity we omit formal definitions of notions of security, and refer the reader to [34]. Roughly speaking, for a scheme that is IND-CPA secure, a ciphertext reveals nothing about the underlying plaintext to an adversary who is given only the ciphertext and the corresponding public key. This is exactly the property that we desire in the FHE setting, where we as the client wish to outsource computation on our encrypted data to an untrusted server, in such a way that the server cannot learn our data. A notion such as *IND-CCA2 security* [193], while stronger, would not make sense in this context, since FHE ciphertexts are malleable by design.

The FV scheme is an augmented version of an encryption scheme due to Lyubashevsky *et al.* [157] and inherits its IND-CPA security from that scheme [100]. In turn, SEAL inherits its IND-CPA security from FV.

We now sketch the security argument in the case of FV. The FV public key is of the form $(p_0, p_1) = (-(as + e), a)$, which is exactly a small secret Ring-LWE sample, so the FV public key is indistinguishable from uniform assuming the hardness of the 1-sample small secret variant of Decision Ring-LWE, and we can say the first component p_0 looks pseudorandom. Now consider the first element of a fresh ciphertext pair, which is of the form $\Delta m + p_0 u + e_1$. This is formed of a term Δm depending deterministically on the message and a masking term $p_0 u + e_1$. Consider the masking term: u is chosen from the Ring-LWE secret distribution, e_1 is from the Ring-LWE error distribution, and by the above argument, p_0 looks pseudorandom. So this the masking term is indistinguishable from a term of the form $a_1 u + e_1$ where a_1 is uniformly random, which is exactly a small secret Ring-LWE sample. Thus the first element of a ciphertext pair is a message term masked by something that looks pseudorandom if Decision Ring-LWE is hard, and hence itself looks pseudorandom. The second element of a ciphertext pair has the form $au + e_2$ where a is uniformly random, u is the same as before and e_2 is from the Ring-LWE error distribution so

n	q	α	usvp	dec	dual
2048	$2^{60} - 2^{14} + 1$	$8.0/q$	125.9	127.4	118.8
4096	$2^{116} - 2^{18} + 1$	$8.0/q$	127.5	125.5	121.7
8192	$2^{226} - 2^{26} + 1$	$8.0/q$	130.0	126.3	124.1
16384	$2^{435} - 2^{33} + 1$	$8.0/q$	135.3	130.7	130.2
32768	$2^{889} - 2^{54} - 2^{53} - 2^{52} + 1$	$8.0/q$	132.8	127.6	127.4

Table 7.2: Default pairs (n, q) in SEAL v2.2 and estimated log of cost of attacks according to commit f13c4a7 of the LWE estimator called with small secret and $m = 2n$ samples.

this is exactly a small secret Ring-LWE sample, and hence also looks pseudorandom if small secret Decision Ring-LWE is hard. We conclude that the entire FV ciphertext is indistinguishable from random assuming the hardness of the 3-sample Decision Ring-LWE problem, or rather the hardness of the 2-sample small secret variant described above, and the assumed uniformity of the public key.

The above argument omits the fact that we also need to make a *weak circular security* assumption [100]. Observe that the evaluation keys, which must be made public so that relinearization can be performed, are essentially maskings of powers of the secret key. Indeed, we can think of the evaluation keys as an encryption of (some power of) the secret key under the secret key itself. Unfortunately, it is not possible to argue the uniformity of the evaluation key based on a Decision Ring-LWE assumption, so we must make the extra assumption that the encryption scheme is secure even when the adversary has access to all of the evaluation keys that may exist.

In order to try to pick parameters in Ring-LWE-based schemes (FHE or otherwise) that we hope are sufficiently secure, we can choose parameters such that the underlying Ring-LWE instance should be hard to solve according to known attacks. Each Ring-LWE sample can be used to extract n LWE samples. To the best of our knowledge, the most powerful attacks against d-sample Ring-LWE all work by instead attacking the nd-sample LWE problem. When estimating the security of a particular set of Ring-LWE parameters we therefore estimate the security of the induced set of LWE parameters. To illustrate this, in Table 7.2 we estimate security of the SEAL default parameters using commit f13c4a7 of the LWE estimator [6].

In FHE schemes based on Ring-LWE, q is typically very large, to enable several homomorphic evaluation operations to be performed before the noise grows too large.

An interesting question is to ask how large q can be before security is lost. The hardness of LWE with q that is exponential in n has been considered in a number of works [185, 45, 145]. Laine and Lauter [145] prove that when q is exponential in n, Babai's Nearest Plane algorithm with LLL as precomputation is sufficient to solve LWE. They implement their attack and observe that performance is better in practice than their proof would suggest. For example, they report that the secret key of an LWE instance with $n = 350$, $q = 2^{52}$ and $\alpha q = 8$ was recovered in approximately 4 days. For comparison, dual attack the same instance would take approximately 2^{28} operations according to the LWE estimator (commit f13c4a7).

Bonnoron and Fontaine [38] ran similar experiments to [145]. They use a decoding approach following [152], using enumeration, in a lattice obtained from an embedding similar to that described by Bai and Galbraith [29]. A basis for the embedding is constructed directly, then is reduced using LLL or BKZ with a small block size, such as 20, and then the enumeration is performed. It is observed that Babai's Nearest Plane is sufficient for the enumeration step in the experiments, and it is argued that this is because the error norm is so small compared to q. The authors conclude that specialised variants of attacks on LWE could be more effective than generic ones in the case of LWE-based FHE, where the error distribution is typically narrow and q is typically large, and call for more research in this area.

7.3 Encoding in Ring-LWE based FHE schemes

In Ring-LWE-based FHE, plaintext elements are very often polynomials in $R_t = \mathbb{Z}_t[x]/(x^n + 1)$ for an integer modulus t. A polynomial plaintext modulus $x - b$ for an integer b is considered for example by Chen *et al.* [61] in the context of the FV scheme [100], but in this thesis we always use an integer plaintext modulus t.

Homomorphic operations on ciphertexts are reflected on the plaintext side as corresponding (multiplication and addition) operations in the ring R_t. In typical applications of homomorphic encryption, the user would want to perform computations not on polynomials but rather on data in the form of integers or rational numbers. The solution is a suitable encoding map, converting the user's inputs to polynomials in R_t; and a corresponding decoding map, converting the resulting plaintext back

into an integer or rational number. In order for the operations on ciphertexts to reflect the operations on the inputs, the encoding and decoding maps need to respect addition and multiplication.

Encoding is a highly non-trivial task that has been considered in a number of works [174, 93, 78, 69, 68, 77]. Clearly since R_t is finite, we can only hope to encode and decode a finite subset of integers. If the subset is large enough, then this would be sufficient in principle. However, there is still a problem, because no non-trivial finite subset of $\mathbb{Z}$ is closed under additions and multiplications, so we have to settle for something that does not respect an arbitrary number of homomorphic operations. Therefore one must be aware of the type of encoding that is used, and perform only operations such that the underlying plaintexts throughout the computation remain possible to decode.

Perhaps the simplest possible encoder is the *scalar encoder*. Given an integer a, we simply encode a as the constant polynomial $a \in R_t$. Obviously we can only encode integers modulo t in this manner. Decoding amounts to reading the constant coefficient of the polynomial and interpreting that as an integer. The problem is that as soon as the underlying plaintext polynomial (constant) wraps around t at any point during the computation, we are no longer doing integer arithmetic, but rather modulo t arithmetic, and decoding might yield an unexpected result. This means that a large t must be chosen, which can be undesirable for noise growth.

The scalar encoder would typically not be used in practice, since it is wasteful in the sense that only one coefficient of a huge polynomial is used to encode and encrypt a small integer. Other encoders, such as the ones we describe below, are more efficient in the sense that many coefficients are used to encode each integer. This means that the size of each nonzero coefficient is smaller than in the scalar encoder case, and hence a smaller t can be chosen. Another method to encode data efficiently is so-called *batching* [205, 43]. This technique, widely used in applications [108, 110, 86, 113], packs many integers into a single plaintext polynomial. The integers are each encoded into a *plaintext slot* and are subsequently operated on in a *Single Instruction Multiple Data* (SIMD) manner.

SEAL implements a family of encoders for encoding integers, parameterised by an

integer base $\beta \geq 2$. The case $\beta = 2$ corresponds to a *binary encoding*. The binary encoder encodes an integer a in the range $[-(2^n - 1), 2^n - 1]$ as follows. It forms the binary expansion of $|a|$, say $a_{n-1} \dots a_1 a_0$, and outputs the polynomial

$$\text{sign}(a) \cdot \left(a_{n-1} x^{n-1} + \dots + a_1 x + a_0\right) .$$

Decoding amounts to evaluating a plaintext polynomial at $x = 2$. When β is set to some integer larger than 2, instead of a binary expansion a balanced base-β expansion is used [93]. Decoding is performed by evaluating a plaintext polynomial at $x = \beta$. Note that with $\beta = 3$ the integer encoder provides encodings with equally small norm as with $\beta = 2$, but with a more compact representation, as it does not waste space in repeating the sign for each non-zero coefficient. Larger β provide even more compact representations, but at the cost of increased coefficients. In most common applications taking $\beta = 2$ or 3 is a good choice, and there is little difference between these two. Recently, Cheon *et al.* [68] showed that using a *Non-Adjacent Form* encoding [197] can be more efficient than a binary encoding or a balanced base-3 encoding.

There are two prominent methods to encode rational numbers. Each has similar performance and limitations, but the methods are not equivalent [59]. One way is to simply scale all rational numbers to integers, encode them using an integer encoder as described above, and record the scaling factor in the clear as a part of the ciphertext. We then need to keep track of the scaling during computations, which results in some inefficiency. As an alternative, SEAL implements a family of *fractional encoders* [93], where such bookkeeping is not required. This family of encoders is again parameterised by a base $\beta \geq 2$, the function of which is exactly the same as in the integer case.

Example 1. *We encode the rational number* 5.8125 *using a fractional encoder with base* $\beta = 2$ *and polynomial modulus* n*. We first express it as a binary expansion*

$$5.8125 = 2^2 + 2^0 + 2^{-1} + 2^{-2} + 2^{-4} .$$

The integer part is encoded as usual with the integer binary encoder, obtaining the polynomial $x^2 + 1$*. For the fractional part, we add* n *to each exponent, and convert it into a polynomial by changing the base* 2 *into the variable* x*. Finally, we flip the signs of each of the terms, in this case obtaining* $-x^{n-1} - x^{n-2} - x^{n-4}$*. The encoding of the fractional part is then added to the encoding of the integer part, to obtain the polynomial* $-x^{n-1} - x^{n-2} - x^{n-4} + x^2 + 1$*.*

In Example 1, we were fortunate that the rational number 5.8125 had a finite binary expansion. Often, the rational number we wish to encode does not have a finite base-β expansion. In this case we allocate some number n_f of the topmost coefficients of the plaintext polynomial coefficients to correspond to the (truncated) fractional part, and reserve the lowest n_i coefficients for the integer part. Homomorphic multiplication will cause the fractional parts of the underlying plaintext polynomials to expand down towards the integer part, and the integer part to expand up towards the fractional part. The decoding process interprets the lowest n_i coefficients as the integer part, and all remaining $n - n_i$ coefficients as belonging to the fractional part, so if the two parts become mixed, decoding will fail.

7.4 Noise growth in SEAL

In this section we give a thorough discussion of the noise growth behaviour of SEAL. This section is based on, and extends, material presented in [146, 58, 60]. We consider both the *inherent noise* and the *invariant noise*, which we now define.

Definition 32. *Let* $\mathtt{ct} = (c_0, c_1, \ldots, c_k)$ *be a ciphertext encrypting the message* $m \in R_t$*. Its* inherent noise *is the unique polynomial* $v_{inh} \in R$ *with smallest infinity norm such that, for some integer coefficient polynomial* a*,*

$$\mathtt{ct}(s) = c_0 + c_1 s + \cdots + c_k s^k = \Delta m + v_{inh} + aq\,.$$

Definition 33. *Let* $\mathtt{ct} = (c_0, c_1, \ldots, c_k)$ *be a ciphertext encrypting the message* $m \in R_t$*. Its* invariant noise *v is the polynomial with the smallest infinity norm such that, for some integer coefficient polynomial a,*

$$\frac{t}{q}\mathtt{ct}(s) = \frac{t}{q}\left(c_0 + c_1 s + \cdots + c_k s^k\right) = m + v + at\,.$$

The two definitions are related in the following way.

Lemma 34. *Let* $\mathtt{ct}$ *be a ciphertext encrypting the message* $m \in R_t$*. The invariant noise v and the inherent noise v_{inh} are related as*

$$v = \frac{t}{q} v_{inh} - \frac{r_t(q)}{q} m\,.$$

Proof: By definition of inherent noise,

$$\begin{aligned}\frac{t}{q}\mathtt{ct}(s) &= \frac{t}{q}\left(c_0 + c_1 s + \cdots + c_k s^k\right)\\ &= \frac{t}{q}\left(\Delta m + v_{inh} + aq\right)\\ &= \frac{q - r_t(q)}{q}m + \frac{t}{q}v_{inh} + at\,,\end{aligned}$$

from which the result follows by definition of invariant noise. □

Once the (invariant or inherent) noise in a ciphertext becomes too large, the ciphertext becomes impossible to decrypt even with the correct secret key. We now state a bound on the inherent noise such that a ciphertext decrypts correctly.

Lemma 35. *A SEAL ciphertext* `ct` *encrypting a message m can be correctly decrypted as long as the inherent noise satisfies $\|v_{inh}\| < \frac{q}{2t} - \frac{t}{2}$.*

Proof: Consider a ciphertext $\mathtt{ct} = (c_0, c_1, \ldots, c_k)$ encrypting the message m under a secret key s. By definition of inherent noise, for some integer coefficient polynomial a, we have $c_0 + c_1 s + \cdots + c_k s^k = \Delta m + v_{inh} + aq$. The decryption function outputs

$$\begin{aligned}m' &= \left[\left\lfloor \frac{t}{q}\left[c_0 + c_1 s + \cdots + c_k s^k\right]_q \right\rceil\right]_t\\ &= \left[\left\lfloor \frac{t}{q}\left[\Delta m + v_{inh} + aq\right]_q \right\rceil\right]_t\\ &= \left[\left\lfloor \frac{t}{q}\left(\Delta m + v_{inh} + a'q\right) \right\rceil\right]_t\\ &= \left[\left\lfloor \frac{t}{q}\left(\Delta m + v_{inh}\right) + a't \right\rceil\right]_t\\ &= \left[\left\lfloor \frac{t}{q}\left(\Delta m + v_{inh}\right) \right\rceil + a't\right]_t\\ &= \left[\left\lfloor \frac{t}{q}\left(\Delta m + v_{inh}\right) \right\rceil\right]_t\\ &= \left[\left\lfloor \frac{q - r_t(q)}{q}m + \frac{t}{q}v_{inh} \right\rceil\right]_t\\ &= \left[\left\lfloor m - \frac{r_t(q)}{q}m + \frac{t}{q}v_{inh} \right\rceil\right]_t\,,\end{aligned}$$

where a' is an integer coefficient polynomial accounting for the reduction modulo q in line 2. This means that $m' = m$ in R_t as long as the terms $-\frac{r_t(q)}{q}m + \frac{t}{q}v_{inh}$ are removed by the rounding. In other words, it suffices that

$$\left\| -\frac{r_t(q)}{q}m + \frac{t}{q}v_{inh} \right\| < \frac{1}{2}\,.$$

If $\|v_{inh}\| < \frac{q}{2t} - \frac{t}{2}$ then

$$\begin{aligned}\left\|-\frac{r_t(q)}{q}m + \frac{t}{q}v_{inh}\right\| &\leq \left\|-\frac{r_t(q)}{q}m\right\| + \left\|\frac{t}{q}v_{inh}\right\| \\ &\leq \frac{r_t(q)}{q}\|m\| + \frac{t}{q}\|v_{inh}\| \\ &< \frac{t}{q}\cdot\frac{t}{2} + \frac{t}{q}\left(\frac{q}{2t} - \frac{t}{2}\right) \\ &\leq \frac{t^2}{2q} + \frac{1}{2} - \frac{t^2}{2q} \\ &= \frac{1}{2}.\end{aligned}$$

□

It is stated in [58, 100] is that the bound $\|v_{inh}\| < \frac{\Delta}{2}$ is sufficient for correct decryption. A detailed proof of this claim was not given in [100]. In [58], the proof only shows that it suffices to require $\|v_{inh}\| < \frac{q}{2t} - t$. It is subsequently shown that $\frac{q}{2t} - t < \frac{\Delta}{2}$, from which the bound $\|v_{inh}\| < \frac{\Delta}{2}$ is incorrectly concluded. We note that the requirement $\|v_{inh}\| < \frac{q}{2t} - t$ is obtained using the bound $\|m\| < t$. Using $\|m\| < \frac{t}{2}$, the same bound $\|v_{inh}\| < \frac{q}{2t} - \frac{t}{2}$ as in Lemma 35 would be obtained.

Note that $\frac{q}{2t} - \frac{t}{2} = \frac{q-t^2}{2t} < \frac{q-t}{2t} \leq \frac{q-r_t(q)}{2t} = \frac{\Delta}{2}$. Therefore proving the bound $\|v_{inh}\| \leq \frac{\Delta}{2}$ would be stronger than the maximal inherent noise bound proved in Lemma 35. Using the same argument as Lemma 35 and the bound $\|v_{inh}\| < \frac{\Delta}{2}$, we can only show

$$\begin{aligned}\left\|-\frac{r_t(q)}{q}m + \frac{t}{q}v_{inh}\right\| &\leq \frac{r_t(q)}{q}\|m\| + \frac{t}{q}\|v_{inh}\| \\ &\leq \frac{t}{q}\cdot\frac{t}{2} + \frac{t}{q}\cdot\frac{\Delta}{2} \\ &= \frac{t^2}{2q} + \frac{q - r_t(q)}{2q} \\ &= \frac{1}{2} + \frac{t^2 - r_t(q)}{2q}.\end{aligned}$$

Since $\frac{t^2-r_t(q)}{2q} > \frac{t^2-t}{2q} > 0$ this does not show that the rounding error is necessarily bounded above by $\frac{1}{2}$.

In the proof of Lemma 35 the goal was to bound the inherent noise such that the rounding procedure in decryption returns a message equal to m modulo t. If the

inherent noise is too large, there will be a rounding error and hence decryption failure. The invariant noise v captures this intuition, since v is exactly the term that is rounded away in the decryption process. We see this in Lemma 36.

Lemma 36. *A SEAL ciphertext* `ct` *encrypting a message m can be correctly decrypted if the invariant noise v satisfies $\|v\| < \frac{1}{2}$.*

Proof: Let $\mathtt{ct} = (c_0, c_1, \ldots, c_k)$. By definition of invariant noise, for some polynomial a with integer coefficients, we have $\frac{t}{q}\left(c_0 + c_1 s + \cdots + c_k s^k\right) = m + v + at$. For some polynomial A with integer coefficients, the decryption function computes:

$$\begin{aligned}
em' &= \left[\left\lfloor \frac{t}{q}\left[c_0 + c_1 s + \cdots + c_k s^k\right]_q \right\rceil\right]_t \\
&= \left[\left\lfloor \frac{t}{q}\left(c_0 + c_1 s + \cdots + c_k s^k + Aq\right) \right\rceil\right]_t \\
&= \left[\left\lfloor \frac{t}{q}\left(c_0 + c_1 s + \cdots + c_k s^k\right) + At \right\rceil\right]_t \\
&= \left[\left\lfloor \frac{t}{q}\left(c_0 + c_1 s + \cdots + c_k s^k\right) \right\rceil + At\right]_t \\
&= \left[\left\lfloor \frac{t}{q}\left(c_0 + c_1 s + \cdots + c_k s^k\right) \right\rceil\right]_t \\
&= \left[\lfloor m + v + at \rceil\right]_t \\
&= \left[m + \lfloor v \rceil\right]_t \ .
\end{aligned}$$

Hence $m' = m$ modulo t if v is removed by the rounding. Therefore decryption is successful if $\|v\| < \frac{1}{2}$. □

The invariant noise in a fresh ciphertext is typically an extremely small fraction, which grows to at most $\frac{1}{2}$ as homomorphic evaluation operations are performed. Such small fractions could be difficult to work with. Moreover, in practice it is more convenient to talk about how much invariant noise we have left, the *invariant noise budget*, until decryption will fail. For example, this enables us to determine if we expect to be able to perform another homomorphic multiplication.

Definition 34. *Let v be the invariant noise of a ciphertext* `ct`*. Then the* invariant noise budget *of* `ct` *is $-\log_2(2\|v\|)$.*

Lemma 37. *A SEAL ciphertext* `ct` *encrypting a message m can be correctly decrypted if the invariant noise budget of* `ct` *is positive.*

As we have seen, if the noise in a ciphertext grows too large then it will not decrypt correctly. It therefore is essential to understand the noise growth behaviour of ciphertexts under homomorphic evaluation operations in order to choose parameters to ensure correctness. In the following subsections we will provide detailed bounds for the inherent and invariant noise growth behaviour for various homomorphic evaluation operations possible in SEAL. Some of the inherent noise bounds involve $r_t(q)$, the remainder of the coefficient modulus q on division by the plaintext modulus t. Since $0 \leq r_t(q) < t$, in the worst case this is equal to $t-1$ (where t may typically be 6 or 8 bits in size). However in practice the parameters q and t are often chosen so that $r_t(q) = 1$, and so the inherent noise growth will be smaller than the bounds would suggest. We use the worst case bound for $r_t(q)$ to highlight the difference between noise growth behaviour when comparing the inherent noise and the invariant noise. However, in the derivations of the noise bounds it should be clear where the $r_t(q)$ terms have been upper bounded by t and so we could produce tighter estimates for the inherent noise growth behaviour if we knew the parameters would be such that $r_t(q)$ was very small.

7.4.1 Inherent noise growth analysis

In this subsection we analyse the inherent noise growth behaviour for the various homomorphic evaluation operations that can be performed in SEAL. We begin with bounding the initial inherent noise in a fresh ciphertext.

Lemma 38 ([100])**.** *Let* $\mathtt{ct} = (c_0, c_1)$ *be a fresh encryption of a message* $m \in R_t$*. The inherent noise* v_{inh} *in* $\mathtt{ct}$ *satisfies*

$$\|v_{inh}\| \leq B(1 + 2\delta)\,.$$

Proof: Let $\mathtt{ct} = (c_0, c_1)$ be a fresh encryption of m under the public key $\mathtt{pk} = (p_0, p_1) = ([-(as + e)]_q, a)$. Then, for some integer coefficient polynomials k_0, k_1, k_2,

$$\begin{aligned} c_0 + c_1 s &= \Delta m + p_0 u + e_1 + k_0 q + p_1 us + e_2 s + k_1 qs \\ &= \Delta m + (-as - e + k_2 q)u + e_1 + aus + e_2 s + k_0 q + k_1 qs \\ &= \Delta m - asu - eu + e_1 + aus + e_2 s + q(k_0 + k_1 s + k_2 u) \\ &= \Delta m - eu + e_1 + e_2 s + q(k_0 + k_1 s + k_2 u)\,, \end{aligned}$$

so the inherent noise is

$$v_{inh} = e_1 - e \cdot u + e_2 \cdot s \,.$$

To bound $\|v_{inh}\|$, we use the fact that the error polynomials sampled from χ have coefficients bounded by B, and that $\|s\| = \|u\| = 1$. Then

$$\|v_{inh}\| \leq B(1 + 2\delta) \,.$$

□

Next we consider the inherent noise growth in a homomorphic addition operation.

Lemma 39. *Let* $\mathtt{ct}_1 = (c_0, c_1, \ldots, c_j)$ *and* $\mathtt{ct}_2 = (d_0, d_1, \ldots, d_k)$ *be two ciphertexts encrypting* $m_1, m_2 \in R_t$*, and having inherent noises* $v_{1_{inh}}$*,* $v_{2_{inh}}$*, respectively. Then the inherent noise* $v_{add_{inh}}$ *in their sum* $\mathtt{ct}_{add}$ *satisfies*

$$\|v_{add_{inh}}\| \leq t + \|v_{1_{inh}}\| + \|v_{2_{inh}}\| \,.$$

Proof: By definition of homomorphic addition, $\mathtt{ct}_{add}$ encrypts $[m_1 + m_2]_t$. Let $[m_1 + m_2]_t = m_1 + m_2 + a_0 t$ for some integer coefficient polynomial a_0. Note that $\|m_1\| < t/2$ and $\|m_2\| < t/2$ so $\|m_1 + m_2\| \leq \|m_1\| + \|m_2\| \leq t$. Therefore $\|a_0\| \leq 1$.

Suppose without loss of generality that $\max(j, k) = j$, so that

$$\mathtt{ct}_{add} = (c_0 + d_0, \ldots, c_k + d_k, c_{k+1}, \ldots c_j) \,.$$

By definition of inherent noise in $\mathtt{ct}_1$ and $\mathtt{ct}_2$, we have

$$\mathtt{ct}_1(s) = \Delta m_1 + v_{1_{inh}} + a_1 q \,,$$
$$\mathtt{ct}_2(s) = \Delta m_2 + v_{2_{inh}} + a_2 q \,,$$

for some polynomials a_1, a_2 with integer coefficients, so

$$\begin{aligned}
\mathtt{ct}_{add}(s) &= (c_0 + d_0) + (c_1 + d_1)s + \cdots + c_j s^j \\
&= c_0 + c_1 \cdot s + \ldots c_j \cdot s^j + d_0 + d_1 \cdot s + \cdots + d_k \cdot s^k \\
&= \mathtt{ct}_1(s) + \mathtt{ct}_2(s) \\
&= \Delta m_1 + v_{1_{inh}} + a_1 q + \Delta m_2 + v_{2_{inh}} + a_2 q \\
&= \Delta(m_1 + m_2) + v_{1_{inh}} + v_{2_{inh}} + (a_1 + a_2)q \\
&= \Delta([m_1 + m_2]_t - a_0 t) + v_{1_{inh}} + v_{2_{inh}} + (a_1 + a_2)q \\
&= \Delta[m_1 + m_2]_t - a_0 \Delta t + v_{1_{inh}} + v_{2_{inh}} + (a_1 + a_2)q \\
&= \Delta[m_1 + m_2]_t - a_0(q - r_t(q)) + v_{1_{inh}} + v_{2_{inh}} + (a_1 + a_2)q \\
&= \Delta[m_1 + m_2]_t + a_0 \cdot r_t(q) + v_{1_{inh}} + v_{2_{inh}} + (a_1 + a_2 - a_0)q\,.
\end{aligned}$$

Thus the inherent noise $v_{add_{inh}} = a_0 \cdot r_t(q) + v_{1_{inh}} + v_{2_{inh}}$. We can bound this inherent noise as follows:

$$\begin{aligned}
\|v_{add_{inh}}\| &= \|a_0 \cdot r_t(q) + v_{1_{inh}} + v_{2_{inh}}\| \\
&\leq \|a_0\| \cdot r_t(q) + \|v_{1_{inh}}\| + \|v_{2_{inh}}\| \\
&\leq r_t(q) + \|v_{1_{inh}}\| + \|v_{2_{inh}}\| \\
&\leq t + \|v_{1_{inh}}\| + \|v_{2_{inh}}\|\,.
\end{aligned}$$

□

Next we consider the inherent noise growth in homomorphic multiplication.

Lemma 40. *Let $\mathtt{ct}_1 = (x_0, \ldots, x_{j_1})$ be a ciphertext of size $j_1 + 1$ encrypting m_1 with inherent noise $v_{1_{inh}}$, and let $\mathtt{ct}_2 = (y_0, \ldots, y_{j_2})$ be a ciphertext of size $j_2 + 1$ encrypting m_2 with inherent noise $v_{2_{inh}}$. Let $J = j_1 + j_2$. Then the inherent noise $v_{mult_{inh}}$ in the product $\mathtt{ct}_{mult}$ satisfies the following bound:*

$$\begin{aligned}
v_{mult_{inh}} \leq\ & \frac{\delta^{J+1} - 1}{2(\delta - 1)} + \frac{t^2 \delta(\delta^{j_2+1} + \delta^{j_1+1} - 2)}{4(\delta - 1)} + \frac{2\delta t^2 + t}{2} - \frac{3\delta t^3}{4q} \\
&+ \left(\delta t + \frac{\delta t(\delta^{j_2+1} - 1)}{2(\delta - 1)} + \frac{\delta t^2}{2q}\right)\|V_1\| + \left(\delta t + \frac{\delta t(\delta^{j_1+1} - 1)}{2(\delta - 1)} + \frac{\delta t^2}{2q}\right)\|V_2\| \\
&+ \left(\frac{3\delta t}{q}\right)\|V_1\| \cdot \|V_2\|\,.
\end{aligned}$$

Proof: By definition of homomorphic multiplication the output ciphertext $\mathtt{ct}_{mult} = (c_0, \ldots c_J)$ is of size $J + 1$ and, for $0 \leq i \leq J$, c_i is such that for some polynomials ϵ_i

with coefficients in $\left(-\frac{1}{2}, \frac{1}{2}\right]$, and for some polynomials A_i with integer coefficients:

$$\begin{aligned} c_i &= \left[\left\lfloor \frac{t}{q}\left(\sum_{k+l=i} x_k y_l\right)\right\rceil\right]_q \\ &= \left\lfloor \frac{t}{q}\left(\sum_{k+l=i} x_k y_l\right)\right\rceil + A_i q \\ &= \frac{t}{q}\left(\sum_{k+l=i} x_k y_l\right) + \epsilon_i + A_i q\,. \end{aligned}$$

For some polynomial b with integer coefficients, $[m_1 m_2]_t = m_1 m_2 + bt$. We can bound $\|b\|$ as follows:

$$\begin{aligned} \|b\| &= \frac{1}{t}\|[m_1 m_2]_t - m_1 m_2\| \\ &\leq \frac{1}{t}\left(\|[m_1 m_2]_t\| + \|m_1 m_2\|\right) \\ &\leq \frac{1}{t}\left(\frac{t}{2} + \delta \cdot \|m_1\| \cdot \|m_2\|\right) \\ &\leq \frac{1}{t}\left(\frac{t}{2} + \delta \cdot \frac{t}{2} \cdot \frac{t}{2}\right) \\ &= \frac{1}{2} + \frac{\delta t}{4}\,. \end{aligned}$$

By definition of inherent noise in $\mathtt{ct}_1$ and $\mathtt{ct}_2$, we have for some polynomials a_1, a_2 with integer coefficients:

$$\begin{aligned} \mathtt{ct}_1(s) &= \Delta m_1 + v_{1_{inh}} + a_1 q \\ \mathtt{ct}_2(s) &= \Delta m_2 + v_{2_{inh}} + a_2 q\,. \end{aligned}$$

For the remainder of the proof, we denote by V_1 the inherent noise $v_{1_{inh}}$ of $\mathtt{ct}_1$ and similarly we denote by V_2 the inherent noise $v_{2_{inh}}$ of $\mathtt{ct}_2$.

By writing $a_i = \frac{1}{q}\left(\mathtt{ct}_i(s) - \Delta m_i - V_i\right)$ and recalling that $\|s\| \leq 1$ we can bound $\|a_i\|$

as follows:

$$\begin{aligned}
\|a_i\| &= \left\| \frac{1}{q} \left(\mathtt{ct}_i(s) - \Delta m_i - V_i \right) \right\| \\
&\leq \frac{1}{q} \left(\|\mathtt{ct}_i(s)\| + \Delta \|m_i\| + \|V_i\| \right) \\
&= \frac{1}{q} \|\mathtt{ct}_i(s)\| + \frac{\Delta}{q} \|m_i\| + \frac{1}{q} \|V_i\| \\
&\leq \frac{1}{q} \|c_0 + c_1 s + \cdots + c_{j_i} s^{j_i}\| + \frac{\Delta t}{2q} + \frac{1}{q} \|V_i\| \\
&\leq \frac{1}{q} \left(\|c_0\| + \delta \cdot \|c_1\| \cdot \|s\| + \cdots + \delta^{j_i} \cdot \|c_{j_i}\| \cdot \|s\|^{j_i} \right) + \frac{q - r_t(q)}{2q} + \frac{1}{q} \|V_i\| \\
&\leq \frac{1}{q} \left(\frac{q}{2} + \delta \cdot \frac{q}{2} \cdot 1 + \cdots + \delta^{j_i} \cdot \frac{q}{2} \cdot 1 \right) + \frac{1}{2} - \frac{r_t(q)}{2q} + \frac{1}{q} \|V_i\| \\
&= \frac{1}{2} \cdot \left(1 + \delta + \cdots + \delta^{j_i} \right) + \frac{1}{2} - \frac{r_t(q)}{2q} + \frac{1}{q} \|V_i\| \\
&= \frac{1}{2} \cdot \frac{\delta^{j_i + 1} - 1}{\delta - 1} + \frac{1}{2} - \frac{r_t(q)}{2q} + \frac{1}{q} \|V_i\| .
\end{aligned}$$

We will also need the bound:

$$\begin{aligned}
\left\| \sum_{i=0}^{J} \epsilon_i s^i \right\| &= \|\epsilon_0 + \epsilon_1 s + \epsilon_2 s^2 + \cdots + \epsilon_J s^J\| \\
&\leq \|\epsilon_0\| + \|\epsilon_1 s\| + \|\epsilon_2 s^2\| + \cdots + \|\epsilon_J s^J\| \\
&\leq \|\epsilon_0\| + \delta \|\epsilon_1\| \cdot \|s\| + \delta^2 \|\epsilon_2\| \|s\|^2 + \cdots + \delta^J \|\epsilon_J\| \|s\|^J \\
&\leq \frac{1}{2} + \delta \cdot \frac{1}{2} \cdot 1 + \delta^2 \cdot \frac{1}{2} \cdot 1 + \cdots + \delta^J \cdot \frac{1}{2} \cdot 1 \\
&= \frac{1}{2} \left(1 + \delta + \delta^2 + \cdots + \delta^J \right) \\
&= \frac{1}{2} \cdot \frac{\delta^{J+1} - 1}{\delta - 1} .
\end{aligned}$$

We can determine the inherent noise $v_{mult_{inh}}$ in $\mathtt{ct}_{mult}$ as follows:

$$\begin{aligned}
\mathtt{ct}_{mult}(s) &= \left(\frac{t}{q}(x_0 y_0) + \epsilon_0 + A_0 q \right) + \cdots + \left(\frac{t}{q}(x_{j_1} y_{j_2}) + \epsilon_J + A_J q \right) s^J \\
&= \frac{t}{q} \sum_{i=0}^{J} \left(\sum_{k+l=i} x_k y_l \right) s^i + \sum_{i=0}^{J} \epsilon_i s^i + q \sum_{i=0}^{J} A_i s^i \\
&= \frac{t}{q} \left(\mathtt{ct}_1(s) \cdot \mathtt{ct}_2(s) \right) + \sum_{i=0}^{J} \epsilon_i s^i + q \sum_{i=0}^{J} A_i s^i \\
&= \frac{t}{q} (\Delta m_1 + V_1 + a_1 q)(\Delta m_2 + V_2 + a_2 q) + \sum_{i=0}^{J} \epsilon_i s^i + q \sum_{i=0}^{J} A_i s^i
\end{aligned}$$

$$
\begin{aligned}
&= \frac{t}{q}\left(\Delta^2 m_1 m_2 + \Delta m_2 V_1 + \Delta m_1 V_2 + V_1 V_2\right) + t a_1 V_2 + t a_2 V_1 \\
&\quad + \Delta t \cdot m_1 a_2 + \Delta t \cdot m_2 a_1 + \sum_{i=0}^{J} \epsilon_i s^i + q\left(\sum_{i=0}^{J} A_i s^i + t a_1 a_2\right) \\
&= \frac{\Delta t}{q} \cdot \Delta m_1 m_2 + \frac{\Delta t}{q} m_2 V_1 + \frac{\Delta t}{q} m_1 V_2 + \frac{t}{q} V_1 V_2 + t a_1 V_2 + t a_2 V_1 \\
&\quad + \Delta t \cdot m_1 a_2 + \Delta t \cdot m_2 a_1 + \sum_{i=0}^{J} \epsilon_i s^i + q\left(\sum_{i=0}^{J} A_i s^i + t a_1 a_2\right) \\
&= \frac{q - r_t(q)}{q} \cdot \Delta m_1 m_2 + \frac{q - r_t(q)}{q} m_2 V_1 + \frac{q - r_t(q)}{q} m_1 V_2 + \frac{t}{q} V_1 V_2 + t a_1 V_2 \\
&\quad + t a_2 V_1 + (q - r_t(q)) \cdot (m_1 a_2 + m_2 a_1) + \sum_{i=0}^{J} \epsilon_i s^i + q\left(\sum_{i=0}^{J} A_i s^i + t a_1 a_2\right) \\
&= \Delta m_1 m_2 - \frac{\Delta \cdot r_t(q)}{q} m_1 m_2 + m_2 V_1 - \frac{r_t(q)}{q} m_2 V_1 + m_1 V_2 - \frac{r_t(q)}{q} m_1 V_2 \\
&\quad + \frac{t}{q} V_1 V_2 + t a_1 V_2 + t a_2 V_1 - r_t(q) m_1 a_2 - r_t(q) m_2 a_1 + \sum_{i=0}^{J} \epsilon_i s^i \\
&\quad + q\left(\sum_{i=0}^{J} A_i s^i + t a_1 a_2 + m_1 a_2 + m_2 a_2\right) \\
&= \Delta [m_1 m_2]_t - \Delta(bt) - \frac{\Delta \cdot r_t(q)}{q} m_1 m_2 + m_2 V_1 - \frac{r_t(q)}{q} m_2 V_1 + m_1 V_2 \\
&\quad - \frac{r_t(q)}{q} m_1 V_2 + \frac{t}{q} V_1 V_2 + t a_1 V_2 + t a_2 V_1 - r_t(q) m_1 a_2 - r_t(q) m_2 a_1 \\
&\quad + \sum_{i=0}^{J} \epsilon_i s^i + q\left(\sum_{i=0}^{J} A_i s^i + t a_1 a_2 + m_1 a_2 + m_2 a_2\right) \\
&= \Delta [m_1 m_2]_t - b(q - r_t(q)) - \frac{\Delta \cdot r_t(q)}{q} m_1 m_2 + m_2 V_1 - \frac{r_t(q)}{q} m_2 V_1 \\
&\quad + m_1 V_2 - \frac{r_t(q)}{q} m_1 V_2 + \frac{t}{q} V_1 V_2 + t a_1 V_2 + t a_2 V_1 - r_t(q) m_1 a_2 \\
&\quad - r_t(q) m_2 a_1 + \sum_{i=0}^{J} \epsilon_i s^i + q\left(\sum_{i=0}^{J} A_i s^i + t a_1 a_2 + m_1 a_2 + m_2 a_2\right) \\
&= \Delta [m_1 m_2]_t + r_t(q) \cdot b - \frac{\Delta \cdot r_t(q)}{q} m_1 m_2 + m_2 V_1 - \frac{r_t(q)}{q} m_2 V_1 + m_1 V_2 \\
&\quad - \frac{r_t(q)}{q} m_1 V_2 + \frac{t}{q} V_1 V_2 + t a_1 V_2 + t a_2 V_1 - r_t(q) m_1 a_2 - r_t(q) m_2 a_1 \\
&\quad + \sum_{i=0}^{J} \epsilon_i s^i + q\left(\sum_{i=0}^{J} A_i s^i + t a_1 a_2 + m_1 a_2 + m_2 a_2 - b\right) .
\end{aligned}
$$

So the inherent noise $v_{mult_{inh}}$ is given by

$$v_{mult_{inh}} = r_t(q) \cdot b - \frac{\Delta \cdot r_t(q)}{q} m_1 m_2 + m_2 V_1 - \frac{r_t(q)}{q} m_2 V_1 + m_1 V_2 - \frac{r_t(q)}{q} m_1 V_2 + \frac{t}{q} V_1 V_2 + t a_1 V_2 + t a_2 V_1 - r_t(q) m_1 a_2 - r_t(q) m_2 a_1 + \sum_{i=0}^{J} \epsilon_i s^i \,.$$

We can bound this inherent noise as follows:

$$\begin{aligned}
\|v_{mult_{inh}}\| &\leq r_t(q) \|b\| + \frac{\Delta \cdot r_t(q)}{q} \|m_1 m_2\| + \|m_2 V_1\| + \frac{r_t(q)}{q} \|m_2 V_1\| + \|m_1 V_2\| \\
&\quad + \frac{r_t(q)}{q} \|m_1 V_2\| + \frac{t}{q} \|V_1 V_2\| + t \|a_1 V_2\| + t \|a_2 V_1\| + r_t(q) \|m_1 a_2\| \\
&\quad + r_t(q) \|m_2 a_1\| + \left\| \sum_{i=0}^{J} \epsilon_i s^i \right\| \\
&\leq r_t(q) \|b\| + \frac{\Delta \cdot r_t(q) \cdot \delta}{q} \|m_1\| \cdot \|m_2\| + \delta \|m_2\| \cdot \|V_1\| \\
&\quad + \frac{r_t(q) \cdot \delta}{q} \|m_2\| \cdot \|V_1\| + \delta \|m_1\| \cdot \|V_2\| + \frac{r_t(q) \cdot \delta}{q} \|m_1\| \cdot \|V_2\| \\
&\quad + \frac{\delta t}{q} \|V_1\| \cdot \|V_2\| + \delta t \|a_1\| \|V_2\| + \delta t \|a_2\| \cdot \|V_1\| \\
&\quad + \delta \cdot r_t(q) \|m_1\| \cdot \|a_2\| + \delta \cdot r_t(q) \|m_2\| \cdot \|a_1\| + \left\| \sum_{i=0}^{J} \epsilon_i s^i \right\| \\
&\leq r_t(q) \|b\| + \frac{\Delta \cdot r_t(q) \cdot \delta \cdot t^2}{4q} + \frac{\delta t}{2} \|V_1\| + \frac{r_t(q) \cdot \delta t}{2q} \cdot \|V_1\| + \frac{\delta t}{2} \cdot \|V_2\| \\
&\quad + \frac{r_t(q) \cdot \delta t}{2q} \cdot \|V_2\| + \frac{\delta t}{q} \|V_1\| \cdot \|V_2\| + \delta t \|a_1\| \|V_2\| + \delta t \|a_2\| \cdot \|V_1\| \\
&\quad + \frac{\delta t \cdot r_t(q)}{2} \cdot \|a_2\| + \frac{\delta t \cdot r_t(q)}{2} \|a_1\| + \left\| \sum_{i=0}^{J} \epsilon_i s^i \right\| \\
&\leq r_t(q) \left(\frac{1}{2} + \frac{\delta t}{4} \right) + \frac{\Delta \cdot r_t(q) \cdot \delta \cdot t^2}{4q} + \frac{\delta t}{2} \|V_1\| + \frac{r_t(q) \cdot \delta t}{2q} \cdot \|V_1\| \\
&\quad + \frac{\delta t}{2} \cdot \|V_2\| + \frac{r_t(q) \cdot \delta t}{2q} \cdot \|V_2\| + \frac{\delta t}{q} \|V_1\| \cdot \|V_2\| \\
&\quad + \delta t \cdot \|V_2\| \left(\frac{\delta^{j_1+1} - 1}{2(\delta - 1)} + \frac{1}{2} - \frac{r_t(q)}{2q} + \frac{1}{q} \|V_1\| \right) \\
&\quad + \delta t \cdot \|V_1\| \left(\frac{\delta^{j_2+1} - 1}{2(\delta - 1)} + \frac{1}{2} - \frac{r_t(q)}{2q} + \frac{1}{q} \|V_2\| \right) \\
&\quad + \frac{\delta t \cdot r_t(q)}{2} \cdot \left(\frac{\delta^{j_2+1} - 1}{2(\delta - 1)} + \frac{1}{2} - \frac{r_t(q)}{2q} + \frac{1}{q} \|V_2\| \right) \\
&\quad + \frac{\delta t \cdot r_t(q)}{2} \left(\frac{\delta^{j_1+1} - 1}{2(\delta - 1)} + \frac{1}{2} - \frac{r_t(q)}{2q} + \frac{1}{q} \|V_1\| \right) + \frac{\delta^{J+1} - 1}{2(\delta - 1)} \\
&\leq \frac{r_t(q)}{2} + \frac{\delta t \cdot r_t(q)}{4} + \frac{\Delta \cdot r_t(q) \cdot \delta \cdot t^2}{4q} + \frac{\delta t}{2} \|V_1\| + \frac{r_t(q) \cdot \delta t}{2q} \cdot \|V_1\|
\end{aligned}$$

$$
\begin{aligned}
&+ \frac{\delta t}{2} \cdot \|V_2\| + \frac{r_t(q) \cdot \delta t}{2q} \cdot \|V_2\| + \frac{\delta t}{q} \|V_1\| \cdot \|V_2\| \\
&+ \left(\frac{t(\delta^{j_1+2} - 1)}{2(\delta - 1)} + \frac{\delta t}{2} - \frac{\delta t \cdot r_t(q)}{2q} \right) \|V_2\| + \frac{\delta t}{q} \|V_1\| \cdot \|V_2\| \\
&+ \left(\frac{t(\delta^{j_2+2} - 1)}{2(\delta - 1)} + \frac{\delta t}{2} - \frac{\delta t \cdot r_t(q)}{2q} \right) \|V_1\| + \frac{\delta t}{q} \|V_2\| \cdot \|v_1\| \\
&+ \frac{\delta t \cdot r_t(q)}{2} \cdot \left(\frac{\delta^{j_2+1} - 1}{2(\delta - 1)} + \frac{1}{2} - \frac{r_t(q)}{2q} \right) + \frac{\delta t \cdot r_t(q)}{2q} \|V_2\| \\
&+ \frac{\delta t \cdot r_t(q)}{2} \left(\frac{\delta^{j_1+1} - 1}{2(\delta - 1)} + \frac{1}{2} - \frac{r_t(q)}{2q} \right) + \frac{\delta t \cdot r_t(q)}{2q} \|V_1\| + \frac{\delta^{J+1} - 1}{2(\delta - 1)} \\
&= \frac{r_t(q)}{2} + \frac{\delta t \cdot r_t(q)}{4} + \frac{\Delta \cdot r_t(q) \cdot \delta \cdot t^2}{4q} + \frac{\delta^{J+1} - 1}{2(\delta - 1)} \\
&+ \frac{\delta t \cdot r_t(q)}{2} \cdot \left(\frac{\delta^{j_2+1} - 1}{2(\delta - 1)} + \frac{1}{2} - \frac{r_t(q)}{2q} + \frac{\delta^{j_1+1} - 1}{2(\delta - 1)} + \frac{1}{2} - \frac{r_t(q)}{2q} \right) \\
&+ \left(\frac{\delta t}{2} + \frac{r_t(q) \cdot \delta t}{2q} + \frac{t(\delta^{j_2+2} - 1)}{2(\delta - 1)} + \frac{\delta t}{2} - \frac{\delta t \cdot r_t(q)}{2q} + \frac{\delta t \cdot r_t(q)}{2q} \right) \|V_1\| \\
&+ \left(\frac{\delta t}{2} + \frac{r_t(q) \cdot \delta t}{2q} + \frac{t(\delta^{j_1+2} - 1)}{2(\delta - 1)} + \frac{\delta t}{2} - \frac{\delta t \cdot r_t(q)}{2q} + \frac{\delta t \cdot r_t(q)}{2q} \right) \|V_2\| \\
&+ \left(\frac{\delta t}{q} + \frac{\delta t}{q} + \frac{\delta t}{q} \right) \|V_1\| \cdot \|V_2\| \\
&= \frac{(3\delta t + 2) \cdot r_t(q)}{4} + \frac{r_t(q) \cdot \delta \cdot t(\Delta t - 2r_t(q))}{4q} \\
&+ \frac{\delta t \cdot r_t(q) \cdot (\delta^{j_2+1} + \delta^{j_1+1} - 2)}{4(\delta - 1)} + \frac{\delta^{J+1} - 1}{2(\delta - 1)} \\
&+ \left(\delta t + \frac{\delta t(\delta^{j_2+1} - 1)}{2(\delta - 1)} + \frac{\delta t \cdot r_t(q)}{2q} \right) \|V_1\| \\
&+ \left(\delta t + \frac{\delta t(\delta^{j_1+1} - 1)}{2(\delta - 1)} + \frac{\delta t \cdot r_t(q)}{2q} \right) \|V_2\| \\
&+ \left(\frac{3\delta t}{q} \right) \|V_1\| \cdot \|V_2\| \\
&= \frac{(3\delta t + 2) \cdot r_t(q)}{4} + \frac{r_t(q) \cdot \delta \cdot t(q - 3r_t(q))}{4q} \\
&+ \frac{\delta t \cdot r_t(q) \cdot (\delta^{j_2+1} + \delta^{j_1+1} - 2)}{4(\delta - 1)} + \frac{\delta^{J+1} - 1}{2(\delta - 1)} \\
&+ \left(\delta t + \frac{\delta t(\delta^{j_2+1} - 1)}{2(\delta - 1)} + \frac{\delta t \cdot r_t(q)}{2q} \right) \|V_1\| \\
&+ \left(\delta t + \frac{\delta t(\delta^{j_1+1} - 1)}{2(\delta - 1)} + \frac{\delta t \cdot r_t(q)}{2q} \right) \|V_2\| \\
&+ \left(\frac{3\delta t}{q} \right) \|V_1\| \cdot \|V_2\|
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{\delta^{J+1}-1}{2(\delta-1)} + \frac{\delta t \cdot r_t(q) \cdot (\delta^{j_2+1} + \delta^{j_1+1} - 2)}{4(\delta-1)} + \frac{(2\delta t+1)\cdot r_t(q)}{2} \\
&\quad - \frac{3\delta t \cdot (r_t(q))^2}{4q} + \left(\delta t + \frac{\delta t(\delta^{j_2+1}-1)}{2(\delta-1)} + \frac{\delta t \cdot r_t(q)}{2q}\right) \|V_1\| \\
&\quad + \left(\delta t + \frac{\delta t(\delta^{j_1+1}-1)}{2(\delta-1)} + \frac{\delta t \cdot r_t(q)}{2q}\right) \|V_2\| + \left(\frac{3\delta t}{q}\right) \|V_1\| \cdot \|V_2\| \\
&\leq \frac{\delta^{J+1}-1}{2(\delta-1)} + \frac{t^2\delta(\delta^{j_2+1} + \delta^{j_1+1} - 2)}{4(\delta-1)} + \frac{2\delta t^2 + t}{2} - \frac{3\delta t^3}{4q} \\
&\quad + \left(\delta t + \frac{\delta t(\delta^{j_2+1}-1)}{2(\delta-1)} + \frac{\delta t^2}{2q}\right) \|V_1\| + \left(\delta t + \frac{\delta t(\delta^{j_1+1}-1)}{2(\delta-1)} + \frac{\delta t^2}{2q}\right) \|V_2\| \\
&\quad + \left(\frac{3\delta t}{q}\right) \|V_1\| \cdot \|V_2\| .
\end{aligned}
$$

□

Next we consider the inherent noise growth in a relinearization operation.

Lemma 41. *Let* $\mathtt{ct}$ *be a ciphertext of size* $M+1$ *encrypting* m*, and having inherent noise* v_{inh}*. Let* $\mathtt{ct}_{relin}$ *of size* $N+1$ *be the ciphertext encrypting* m*, obtained by the relinearization of* $\mathtt{ct}$*, where* $2 \leq N+1 < M+1$*. Then, the inherent noise* $v_{relin_{inh}}$ *in* $\mathtt{ct}_{relin}$ *is given by*

$$
v_{relin_{inh}} = v_{inh} - \sum_{j=0}^{M-N-1} \sum_{i=0}^{\ell} e_{M-j,i} c_{M-j}^{(i)} ,
$$

and can be bounded as

$$
\|v_{relin_{inh}}\| \leq \|v_{inh}\| + (M-N)(\ell+1)\delta B w .
$$

Proof: The relinearization of a ciphertext from size $M+1$ to size $N+1$, where $2 \leq N+1 < M+1$, consists of $M-N$ *one-step* relinearizations. In each step, the *current* ciphertext $(c_0, c_1, \ldots, c_k)$ is transformed to an intermediate ciphertext $\mathtt{ct}' = (c'_0, c'_1, \ldots, c'_{k-1})$ using the appropriate evaluation key

$$
\mathtt{evk}_k = [([-(a_{k,i}s + e_{k,i}) + w^i s^k]_q, a_{k,i}) : i = 0, \ldots, \ell] .
$$

In the following step, $\mathtt{ct}'$ becomes the current ciphertext, and so on until the intermediate ciphertext produced is of size $N+1$, at which point it is output as $\mathtt{ct}_{relin}$. By definition of the inherent noise in the input ciphertext we have for some integer coefficient polynomial a:

$$
\mathtt{ct}(s) = c_0 + c_1 s + \cdots + c_M s^M = \Delta m + v_{inh} + aq .
$$

The input ciphertext is $\mathtt{ct} = (c_0, c_1, \ldots, c_M)$, and after the first one-step relinearization, the intermediate ciphertext is $\mathtt{ct}' = (c'_0, c'_1, \ldots, c'_{M-1})$, where

$$c'_0 = c_0 + \sum_{i=0}^{\ell} \mathtt{evk}_M[i][0] c_M^{(i)}, \quad c'_1 = c_1 + \sum_{i=0}^{\ell} \mathtt{evk}_M[i][1] c_M^{(i)},$$

and $c'_j = c_j$ for $2 \leq j \leq M-1$. So, for some polynomials a_i with integer coefficients, where $0 \leq i \leq \ell + 1$:

$$\begin{aligned}
\mathtt{ct}'(s) &= \left(c'_0 + c'_1 s + \ldots + c'_{M-1} s^{M-1}\right) \\
&= c_0 + \sum_{i=0}^{\ell} \mathtt{evk}_M[i][0] c_M^{(i)} + \left(c_1 + \sum_{i=0}^{\ell} \mathtt{evk}_M[i][1] c_M^{(i)}\right) s + \ldots + c_{M-1} s^{M-1} \\
&= \sum_{i=0}^{\ell} \mathtt{evk}_M[i][0] c_M^{(i)} + s \sum_{i=0}^{\ell} \mathtt{evk}_M[i][1] c_M^{(i)} + c_0 + c_1 s + \ldots + c_{M-1} s^{M-1} \\
&= -\sum_{i=0}^{\ell} e_{M,i} c_M^{(i)} + \sum_{i=0}^{\ell} a_i q c_M^{(i)} + s^M \sum_{i=0}^{\ell} w^i c_M^{(i)} + c_0 + c_1 s + \ldots + c_{M-1} s^{M-1} \\
&= -\sum_{i=0}^{\ell} e_{M,i} c_M^{(i)} + \sum_{i=0}^{\ell} a_i q c_M^{(i)} + s^M c_M + c_0 + c_1 s + \ldots + c_{M-1} s^{M-1} \\
&= -\sum_{i=0}^{\ell} e_{M,i} c_M^{(i)} + c_0 + c_1 s + \ldots + c_{M-1} s^{M-1} + s^M c_M + q \sum_{i=0}^{\ell} a_i c_M^{(i)} \\
&= -\sum_{i=0}^{\ell} e_{M,i} c_M^{(i)} + \Delta m + v_{inh} + aq + q \sum_{i=0}^{\ell} a_i c_M^{(i)} \\
&= \Delta m + v_{inh} - \sum_{i=0}^{\ell} e_{M,i} c_M^{(i)} + q \left(a + \sum_{i=0}^{\ell} a_i c_M^{(i)}\right).
\end{aligned}$$

Hence, the noise grows by an additive factor $-\sum_{i=0}^{\ell} e_{M,i} c_M^{(i)}$ in a one-step relinearization. Iterating this process, we find the noise after relinearization:

$$v_{relin_{inh}} = v_{inh} - \sum_{j=0}^{M-N-1} \sum_{i=0}^{\ell} e_{M-j,i} c_{M-j}^{(i)}.$$

We can bound $\|v_{relin_{inh}}\|$ as follows:

$$\begin{aligned}
\|v_{relin_{inh}}\| &\leq \|v_{inh}\| + \sum_{j=0}^{M-N-1} \sum_{i=0}^{\ell} \left\| e_{M-j,i} c_{M-j}^{(i)} \right\| \\
&\leq \|v_{inh}\| + (M-N)(\ell+1)\delta B w.
\end{aligned}$$

□

Next we consider the inherent noise growth in a plain multiplication operation.

Lemma 42. *Let* $\mathtt{ct}$ *be a ciphertext encrypting* m_1 *with inherent noise* v_{inh} *and let* m_2 *be a plaintext. Let* $\mathtt{ct}_{multp}$ *denote the ciphertext obtained by plain multiplication of* $\mathtt{ct}$ *with* m_2*. Then the inherent noise* $v_{multp_{inh}}$ *in* $\mathtt{ct}_{multp}$ *is bounded as follows:*

$$\|v_{multp_{inh}}\| \leq \frac{t}{2} + \frac{\delta t^2}{4} + \frac{\delta t}{2}\|v_{inh}\|\,.$$

Proof: By definition of plain multiplication, $\mathtt{ct}_{multp} = (m_2c_0, m_2c_1, \ldots, m_2c_k)$. For some polynomial b with integer coefficients, $[m_1m_2]_t = m_1m_2 + bt$. We can then bound $\|b\|$ as follows:

$$\begin{aligned}
\|b\| &= \frac{1}{t}\|[m_1m_2]_t - m_1m_2\| \\
&\leq \frac{1}{t}\left(\|[m_1m_2]_t\| + \|m_1m_2\|\right) \\
&\leq \frac{1}{t}\left(\frac{t}{2} + \delta \cdot \|m_1\|\|m_2\|\right) \\
&= \frac{1}{2} + \frac{\delta}{2}\|m_2\|\,.
\end{aligned}$$

We have for some polynomial a with integer coefficients:

$$\begin{aligned}
\mathtt{ct}_{multp}(s) &= c_0m_2 + c_1m_2s + \cdots + c_km_2s^k \\
&= m_2\mathtt{ct}(s) \\
&= \Delta m_1m_2 + m_2v_{inh} + m_2aq \\
&= \Delta[m_1m_2]_t - (\Delta t)b + m_2v_{inh} + m_2aq \\
&= \Delta[m_1m_2]_t + r_t(q)b + m_2v_{inh} + (m_2a - b)q\,.
\end{aligned}$$

Hence the inherent noise is

$$v_{multp_{inh}} = r_t(q) \cdot b + m_2 \cdot v_{inh}\,.$$

Thus we can bound this inherent noise as follows:

$$\begin{aligned}
\|v_{multp_{inh}}\| &= \|r_t(q) \cdot b + m_2 \cdot v_{inh}\| \\
&\leq r_t(q)\left(\frac{1}{2} + \frac{\delta}{2}\|m_2\|\right) + \delta \cdot \|m_2\| \cdot \|v_{inh}\| \\
&\leq \frac{t}{2} + \frac{\delta t}{2}\|m_2\| + \delta \cdot \|m_2\| \cdot \|v_{inh}\| \\
&\leq \frac{t}{2} + \frac{\delta t^2}{4} + \frac{\delta t}{2}\|v_{inh}\|\,.
\end{aligned}$$

□

Next we consider the inherent noise growth in a plain addition operation.

Lemma 43. *Let* $\mathtt{ct} = (c_0, c_1, \ldots, c_k)$ *be a ciphertext encrypting* m_1 *with inherent noise* v_{inh} *and* m_2 *be a plaintext. Then the result* $\mathtt{ct}_{addp}$ *of plain addition of* $\mathtt{ct}$ *and* m_2 *has inherent noise* $v_{addp_{inh}}$ *such that*

$$\|v_{addp_{inh}}\| \leq \|v_{inh}\| + t\,.$$

Proof: By definition of plain addition $\mathtt{ct}_{addp} = (c_0 + \Delta m_2, c_1, \ldots, c_k)$. By definition of inherent noise, $\mathtt{ct}(s) = \Delta m_1 + v_{inh} + aq$ for some integer coefficient polynomial a. Let $[m_1 + m_2]_t = m_1 + m_2 + bt$ for some integer coefficient polynomial b. Note that $\|m_1\| < t/2$ and $\|m_2\| < t/2$ so $\|m_1 + m_2\| \leq \|m_1\| + \|m_2\| \leq t$. Therefore $\|b\| \leq 1$, and

$$\begin{aligned}
\mathtt{ct}_{addp}(s) &= c_0 + \Delta m_2 + c_1 s + \cdots + c_k s^k \\
&= \Delta m_2 + \Delta m_1 + v_{inh} + aq \\
&= \Delta([m_1 + m_2]_t - bt) + v_{inh} + aq \\
&= \Delta[m_1 + m_2]_t - (q - r_t(q))b + v_{inh} + aq \\
&= \Delta[m_1 + m_2]_t + r_t(q) \cdot b + v_{inh} + (a - b)q\,.
\end{aligned}$$

So the inherent noise $v_{addp_{inh}} = r_t(q) \cdot b + v_{inh}$. We can then bound this inherent noise as follows:

$$\|v_{addp_{inh}}\| \leq \|v_{inh}\| + r_t(q) \cdot \|b\| \leq \|v_{inh}\| + t\,.$$

□

Next we consider the inherent noise growth in a homomorphic negation operation.

Lemma 44. *Let* $\mathtt{ct}$ *be a ciphertext encrypting* m *with inherent noise* v_{inh} *and* $\mathtt{ct}_{neg}$ *be its negation. The inherent noise* $v_{neg_{inh}}$ *in* $\mathtt{ct}_{neg_{inh}}$ *satisfies*

$$\|v_{neg_{inh}}\| \leq \|v_{inh}\| + t\,.$$

Proof: If $\mathtt{ct} = (c_0, c_1, \ldots, c_k)$ then its negation $\mathtt{ct}_{neg} = (-c_0, -c_1, \ldots, -c_k)$. Note that since $\|m\| < t/2$ we have $[-m]_t = -m + bt$ for $\|b\| \leq 1$. Thus for some integer

coefficient polynomial a:

$$\begin{aligned}\mathtt{ct}_{neg}(s) &= -(c_0, c_1, \ldots, c_k) \\ &= -(\Delta m + v_{inh} + aq) \\ &= \Delta([-m]_t - bt) - v_{inh} - aq \\ &= \Delta[-m]_t - (q - r_t(q))b - v_{inh} - aq \\ &= \Delta[-m]_t + r_t(q) \cdot b - v_{inh} - (b + a)q\,.\end{aligned}$$

Hence the noise $v_{neg_{inh}} = r_t(q) \cdot b - v_{inh}$ and can be bounded as

$$\begin{aligned}\|v_{neg_{inh}}\| &= \|r_t(q) \cdot b - v_{inh}\| \\ &\leq r_t(q) \cdot \|b\| + \|v_{inh}\| \\ &\leq t + \|v_{inh}\|\,.\end{aligned}$$

□

Finally, we consider the inherent noise growth in a homomorphic subtraction.

Lemma 45. *Let $\mathtt{ct}_1 = (c_0, c_1, \ldots, c_j)$ and $\mathtt{ct}_2 = (d_0, d_1, \ldots, d_k)$ be two ciphertexts encrypting $m_1, m_2 \in R_t$, and having inherent noises $v_{1_{inh}}$, $v_{2_{inh}}$, respectively. Then the inherent noise $v_{sub_{inh}}$ in the ciphertext $\mathtt{ct}_{sub}$ obtained as the result of homomorphic subtraction satisfies*

$$\|v_{sub_{inh}}\| \leq t + \|v_{1_{inh}}\| + \|v_{2_{inh}}\|\,.$$

Proof: By definition of homomorphic subtraction, $\mathtt{ct}_{sub}$ encrypts $[m_1 - m_2]_t$. Let $[m_1 - m_2]_t = m_1 - m_2 + a_0 t$ for some integer coefficient polynomial a_0. Note that $\|m_1\| < t/2$ and $\|m_2\| < t/2$ so $\|m_1 - m_2\| \leq \|m_1\| + \| - m_2\| = \|m_1\| + \|m_2\| \leq t$. Therefore $\|a_0\| \leq 1$.

Suppose without loss of generality that $\max(j, k) = j$, so that

$$\mathtt{ct}_{sub} = (c_0 - d_0, \ldots, c_k - d_k, c_{k+1}, \ldots c_j)\,.$$

By definition of inherent noise in $\mathtt{ct}_1$ and $\mathtt{ct}_2$, we have

$$\begin{aligned}\mathtt{ct}_1(s) &= \Delta m_1 + v_{1_{inh}} + a_1 q\,, \\ \mathtt{ct}_2(s) &= \Delta m_2 + v_{2_{inh}} + a_2 q\,,\end{aligned}$$

for some polynomials a_1, a_2 with integer coefficients, so

$$\begin{aligned}
\mathtt{ct}_{sub}(s) &= (c_0 - d_0) + (c_1 - d_1)s + \cdots + c_j s^j \\
&= \mathtt{ct}_1(s) - \mathtt{ct}_2(s) \\
&= \Delta m_1 + v_{1_{inh}} + a_1 q - (\Delta m_2 + v_{2_{inh}} + a_2 q) \\
&= \Delta([m_1 - m_2]_t - a_0 t) + v_{1_{inh}} - v_{2_{inh}} + (a_1 - a_2)q \\
&= \Delta[m_1 - m_2]_t - (q - r_t(q))a_0 + v_{1_{inh}} - v_{2_{inh}} + (a_1 - a_2)q \\
&= \Delta[m_1 - m_2]_t + r_t(q) \cdot a_0 + v_{1_{inh}} - v_{2_{inh}} + (a_1 - a_2 - a_0)q\,.
\end{aligned}$$

We conclude that the inherent noise $v_{sub_{inh}} = r_t(q) \cdot a_0 + v_{1_{inh}} + v_{2_{inh}}$. We can bound this inherent noise as follows:

$$\begin{aligned}
\|v_{sub_{inh}}\| &= \|r_t(q) \cdot a_0 + v_{1_{inh}} + v_{2_{inh}}\| \\
&\leq r_t(q) \cdot \|a_0\| + \|v_{1_{inh}}\| + \|v_{2_{inh}}\| \\
&\leq t + \|v_{1_{inh}}\| + \|v_{2_{inh}}\|\,.
\end{aligned}$$

□

7.4.2 Invariant noise growth analysis

In this subsection we analyse the invariant noise growth behaviour for the various homomorphic evaluation operations that can be performed in SEAL. We begin with bounding the initial invariant noise in a fresh ciphertext.

Lemma 46. *Let* $\mathtt{ct} = (c_0, c_1)$ *be a fresh encryption of a message* $m \in R_t$*. The invariant noise* v *in* $\mathtt{ct}$ *satisfies*

$$\|v\| \leq \frac{t}{q}\|m\| + \frac{tB}{q}(2\delta + 1)\,.$$

Proof: Let $\mathtt{ct} = (c_0, c_1)$ be a fresh encryption of m under the public key $\mathtt{pk} =$

$(p_0, p_1) = ([-(as + e)]_q, a)$. Then, for some integer coefficient polynomials k_0, k_1, k,

$$\begin{aligned}\frac{t}{q}(c_0 + c_1 s) &= \frac{t}{q}(\Delta m + p_0 u + e_0 + k_0 q + p_1 u s + e_1 s + k_1 q s) \\ &= \frac{q - r_t(q)}{q} m + \frac{t}{q}(p_0 u + e_1 + p_1 u s + e_2 s) + t(k_1 + k_2 s) \\ &= m + \frac{t}{q}\left(\frac{-r_t(q) m}{t} + p_0 u + e_0 + p_1 u s + e_1 s\right) + t(k_0 + k_1 s) \\ &= m + \frac{t}{q}\left(\frac{-r_t(q)}{t} m - e u + e_1 + e_2 s\right) + t(k_0 + k_1 s + k u)\,,\end{aligned}$$

so the invariant noise is

$$v = \frac{t}{q}\left(\frac{-r_t(q)}{t} m - e u + e_1 + e_2 s\right).$$

To bound $\|v\|$, we use the fact that the error polynomials sampled from χ have coefficients bounded by B, and that $\|s\| = \|u\| = 1$. Then

$$\begin{aligned}\|v\| &= \left\| \frac{t}{q}\left(\frac{-r_t(q)}{t} m - e u + e_1 + e_2 s\right)\right\| \\ &\leq \frac{r_t(q)}{q}\|m\| + \frac{t}{q}(\delta\|e\|\|u\| + \|e_1\| + \delta\|e_2\|\|s\|) \\ &\leq \frac{t}{q}\|m\| + \frac{tB}{q}(2\delta + 1)\,.\end{aligned}$$

□

Next we consider the invariant noise growth in a homomorphic addition operation.

Lemma 47. *Let $\mathtt{ct}_1 = (c_0, c_1, \ldots, c_j)$ and $\mathtt{ct}_2 = (d_0, d_1, \ldots, d_k)$ be two ciphertexts encrypting $m_1, m_2 \in R_t$, and having noises v_1, v_2, respectively. Then the noise v_{add} in their sum $\mathtt{ct}_{add}$ is $v_{add} = v_1 + v_2$, and satisfies*

$$\|v_{add}\| \leq \|v_1\| + \|v_2\|\,.$$

Proof: By definition of homomorphic addition, $\mathtt{ct}_{add}$ encrypts $[m_1 + m_2]_t$. Let $[m_1 + m_2]_t = m_1 + m_2 + a_0 t$ for some integer coefficient polynomial a_0. Suppose without loss of generality that $\max(j, k) = j$, so that

$$\mathtt{ct}_{add} = (c_0 + d_0, \ldots, c_k + d_k, c_{k+1}, \ldots c_j)\,.$$

By definition of invariant noise in $\mathtt{ct}_1$ and $\mathtt{ct}_2$, we have

$$\frac{t}{q}\mathtt{ct}_1(s) = m_1 + v_1 + a_1 t\,,$$
$$\frac{t}{q}\mathtt{ct}_2(s) = m_2 + v_2 + a_2 t\,,$$

for some polynomials a_1, a_2 with integer coefficients. Thus:

$$\begin{aligned}\frac{t}{q}\mathtt{ct}_{add}(s) &= \frac{t}{q}\left((c_0+d_0) + (c_1+d_1)s + \cdots + c_j s^j\right)\\ &= \frac{t}{q}\mathtt{ct}_1(s) + \frac{t}{q}\mathtt{ct}_2(s)\\ &= [m_1+m_2]_t + v_1 + v_2 + (a_1+a_2-a_0)t\,,\end{aligned}$$

so the noise is $v_{add} = v_1 + v_2$, and $\|v_{add}\| = \|v_1+v_2\| \le \|v_1\| + \|v_2\|$. □

Next we consider the invariant noise growth in homomorphic multiplication.

Lemma 48. *Let $\mathtt{ct}_1 = (x_0, \ldots, x_{j_1})$ be a ciphertext of size j_1+1 encrypting m_1 with noise v_1, and let $\mathtt{ct}_2 = (y_0, \ldots, y_{j_2})$ be a ciphertext of size j_2+1 encrypting m_2 with noise v_2. Then the invariant noise v_{mult} in their product $\mathtt{ct}_{mult}$ satisfies the following bound:*

$$\|v_{mult}\| \le \left[\delta t + \left(\frac{\delta t}{2}\cdot\frac{\delta^{j_2+1}-1}{\delta-1}\right)\right]\|v_1\| + \left[\delta t + \left(\frac{\delta t}{2}\cdot\frac{\delta^{j_1+1}-1}{\delta-1}\right)\right]\|v_2\| + 3\delta\cdot\|v_1\|\cdot\|v_2\| + \frac{t(\delta^{J+1}-1)}{2q(\delta-1)}\,.$$

Proof: By the definition of homomorphic multiplication the product ciphertext $\mathtt{ct}_{mult} = (c_0, \ldots c_J)$ is of size $J+1$ where $J = j_1 + j_2$ and, for $0 \le i \le J$, c_i is such that for some polynomials ϵ_i with coefficients in $\left(-\frac{1}{2}, \frac{1}{2}\right]$, and for some polynomials A_i with integer coefficients,

$$c_i = \left[\left\lfloor \frac{t}{q}\left(\sum_{k+l=i} x_k y_l\right)\right\rceil\right]_q = \left\lfloor \frac{t}{q}\left(\sum_{k+l=i} x_k y_l\right)\right\rceil + A_i q = \frac{t}{q}\left(\sum_{k+l=i} x_k y_l\right) + \epsilon_i + A_i q\,.$$

For some polynomial b with integer coefficients, $[m_1 m_2]_t = m_1 m_2 + bt$. By definition of invariant noise in $\mathtt{ct}_1$ and $\mathtt{ct}_2$, we have for some polynomials a_1, a_2 with integer coefficients,

$$\frac{t}{q}\mathtt{ct}_1(s) = m_1 + v_1 + a_1 t$$
$$\frac{t}{q}\mathtt{ct}_2(s) = m_2 + v_2 + a_2 t\,.$$

We write $a_i t = \frac{t}{q}\mathtt{ct}_i(s) - m_i - v_i$ and using $\|s\| \leq 1$ we note that:

$$\begin{aligned}
\|a_i t\| &= \left\| \frac{t}{q}\mathtt{ct}_i(s) - m_i - v_i \right\| \\
&\leq \frac{t}{q}\|\mathtt{ct}_i(s)\| + \|m_i\| + \|v_i\| \\
&\leq \frac{t}{q}\left(\|c_0\| + \delta \cdot \|c_1\| \cdot \|s\| + \cdots + \delta^{j_i} \cdot \|c_{j_i}\| \cdot \|s\|^{j_i}\right) + \|m_i\| + \|v_i\| \\
&\leq \frac{t}{q}\left(\frac{q}{2} + \delta \cdot \frac{q}{2} \cdot 1 + \cdots + \delta^{j_i} \cdot \frac{q}{2} \cdot 1\right) + \|m_i\| + \|v_i\| \\
&= \frac{t}{2} \cdot \frac{\delta^{j_i+1} - 1}{\delta - 1} + \|m_i\| + \|v_i\| .
\end{aligned}$$

We can also bound:

$$\begin{aligned}
\left\| \sum_{i=0}^{J} \epsilon_i s^i \right\| &= \|\epsilon_0 + \epsilon_1 s + \epsilon_2 s^2 + \cdots + \epsilon_J s^J\| \\
&\leq \|\epsilon_0\| + \delta\|\epsilon_1\| \cdot \|s\| + \delta^2\|\epsilon_2\|\|s\|^2 + \cdots + \delta^J\|\epsilon_J\|\|s\|^J \\
&\leq \frac{1}{2} + \delta \cdot \frac{1}{2} \cdot 1 + \delta^2 \cdot \frac{1}{2} \cdot 1 + \cdots + \delta^J \cdot \frac{1}{2} \cdot 1 \\
&= \frac{1}{2} \cdot \frac{\delta^{J+1} - 1}{\delta - 1} .
\end{aligned}$$

We can determine the invariant noise v_{mult} in $\mathtt{ct}_{mult}$ as follows:

$$\begin{aligned}
\frac{t}{q}\mathtt{ct}_{mult}(s) &= \frac{t}{q}(c_0, \ldots, c_J)(s) \\
&= \frac{t}{q}\left[\frac{t}{q}\sum_{i=0}^{J}\left(\sum_{k+l=i} x_k y_l\right) s^i + \sum_{i=0}^{J} \epsilon_i s^i\right] + \left(\sum_{i=0}^{J} A_i s^i\right) t \\
&= \frac{t}{q} \cdot \frac{t}{q}\left[\sum_{i=0}^{J}\left(\sum_{k+l=i} x_k y_l\right) s^i\right] + \frac{t}{q}\sum_{i=0}^{J} \epsilon_i s^i + \left(\sum_{i=0}^{J} A_i s^i\right) t \\
&= \frac{t}{q}\mathtt{ct}_1(s) \cdot \frac{t}{q}\mathtt{ct}_2(s) + \frac{t}{q}\sum_{i=0}^{J} \epsilon_i s^i + \left(\sum_{i=0}^{J} A_i s^i\right) t \\
&= (m_1 + v_1 + a_1 t)(m_2 + v_2 + a_2 t) + \frac{t}{q}\sum_{i=0}^{J} \epsilon_i s^i + \left(\sum_{i=0}^{J} A_i s^i\right) t \\
&= [m_1 m_2]_t + m_1 v_2 + m_2 v_1 + v_1 v_2 + v_1 a_2 t + v_2 a_1 t + \frac{t}{q}\sum_{i=0}^{J} \epsilon_i s^i \\
&\quad + \left(m_1 a_2 + m_2 a_1 + a_1 a_2 t + \sum_{i=0}^{J} A_i s^i - b\right) t .
\end{aligned}$$

Thus the invariant noise is given by

$$v_{mult} = m_1 v_2 + m_2 v_1 + v_1 v_2 + v_1 a_2 t + v_2 a_1 t + \frac{t}{q} \sum_{i=0}^{j_1+j_2} \epsilon_i s^i \,.$$

We can now bound the invariant noise as follows:

$$\begin{aligned}
\|v_{mult}\| &= \left\| m_1 v_2 + m_2 v_1 + v_1 v_2 + v_1 a_2 t + v_2 a_1 t + \frac{t}{q} \sum_{i=0}^{J} \epsilon_i s^i \right\| \\
&\leq \|m_1 v_2\| + \|m_2 v_1\| + \|v_1 v_2\| + \|v_1 a_2 t\| + \|v_2 a_1 t\| + \frac{t}{q} \left\| \sum_{i=0}^{J} \epsilon_i s^i \right\| \\
&\leq \delta \cdot \frac{t}{2} \cdot \|v_2\| + \delta \cdot \frac{t}{2} \cdot \|v_1\| + \delta \cdot \|v_1\| \cdot \|v_2\| \\
&\quad + \delta \cdot \|v_1\| \cdot \left(\frac{t}{2} \cdot \frac{\delta^{j_2+1} - 1}{\delta - 1} + \|m_2\| + \|v_2\| \right) \\
&\quad + \delta \cdot \|v_2\| \cdot \left(\frac{t}{2} \cdot \frac{\delta^{j_1+1} - 1}{\delta - 1} + \|m_1\| + \|v_1\| \right) + \frac{t}{q} \cdot \frac{1}{2} \cdot \frac{\delta^{J+1} - 1}{\delta - 1} \\
&\leq \delta \cdot \frac{t}{2} \cdot \|v_2\| + \delta \cdot \frac{t}{2} \cdot \|v_1\| + \delta \cdot \|v_1\| \cdot \|v_2\| + \frac{t}{q} \cdot \frac{1}{2} \cdot \frac{\delta^{J+1} - 1}{\delta - 1} \\
&\quad + \delta \cdot \|v_1\| \cdot \left(\frac{t}{2} \cdot \frac{\delta^{j_2+1} - 1}{\delta - 1} \right) + \delta \cdot \|v_1\| \cdot \|m_2\| + \delta \cdot \|v_1\| \cdot \|v_2\| \\
&\quad + \delta \cdot \|v_2\| \cdot \left(\frac{t}{2} \cdot \frac{\delta^{j_1+1} - 1}{\delta - 1} \right) + \delta \cdot \|v_2\| \cdot \|m_1\| + \delta \cdot \|v_2\| \cdot \|v_1\| \\
&\leq \left[\delta t + \left(\frac{\delta t}{2} \cdot \frac{\delta^{j_2+1} - 1}{\delta - 1} \right) \right] \|v_1\| + \left[\delta t + \left(\frac{\delta t}{2} \cdot \frac{\delta^{j_1+1} - 1}{\delta - 1} \right) \right] \|v_2\| \\
&\quad + 3\delta \cdot \|v_1\| \cdot \|v_2\| + \frac{t(\delta^{J+1} - 1)}{2q(\delta - 1)} \,.
\end{aligned}$$

□

Next we consider the invariant noise growth in a relinearization operation.

Lemma 49. *Let* $\mathtt{ct}$ *be a ciphertext of size* $M+1$ *encrypting* m*, and having noise* v*. Let* $\mathtt{ct}_{relin}$ *of size* $N+1$ *be the ciphertext encrypting* m*, obtained by the relinearization of* $\mathtt{ct}$*, where* $2 \leq N+1 < M+1$*. Then, the noise* v_{relin} *in* $\mathtt{ct}_{relin}$ *is given by*

$$v_{relin} = v - \frac{t}{q} \sum_{j=0}^{M-N-1} \sum_{i=0}^{\ell} e_{(M-j),i} c_{M-j}^{(i)} \,,$$

and can be bounded as

$$\|v_{relin}\| \leq \|v\| + \frac{t}{q}(M-N)\delta B(\ell+1)w \,.$$

Proof: The relinearization of a ciphertext from size $M+1$ to size $N+1$, where $2 \leq N+1 < M+1$, consists of $M-N$ one-step relinearizations. In each step, the current ciphertext $(c_0, c_1, \ldots, c_k)$ is transformed to an intermediate ciphertext $\mathtt{ct}' = (c'_0, c'_1, \ldots, c'_{k-1})$ using the appropriate evaluation key

$$\mathtt{evk}_k = [([-(a_{k,i}s + e_{k,i}) + w^i s^k]_q, a_{k,i}) : i = 0, \ldots, \ell]\,.$$

In the following step, $\mathtt{ct}'$ becomes the current ciphertext, and so on until the intermediate ciphertext produced is of size $N+1$, at which point it is output as $\mathtt{ct}_{relin}$. By definition of the invariant noise in the input ciphertext, we have for some integer coefficient polynomial $a_{\ell+1}$,

$$\frac{t}{q}\mathtt{ct}(s) = \frac{t}{q}\left(c_0 + c_1 s + \cdots + c_M s^M\right) = m + v + a_{\ell+1}t\,.$$

The input ciphertext is $\mathtt{ct} = (c_0, c_1, \ldots, c_M)$, and after the first one-step relinearization, the intermediate ciphertext is $\mathtt{ct}' = (c'_0, c'_1, \ldots, c'_{M-1})$, where

$$c'_0 = c_0 + \sum_{i=0}^{\ell} \mathtt{evk}_M[i][0]c_M^{(i)}\,, \quad c'_1 = c_1 + \sum_{i=0}^{\ell} \mathtt{evk}_M[i][1]c_M^{(i)}\,,$$

and $c'_j = c_j$ for $2 \leq j \leq M-1$. So, for some polynomials a_i with integer coefficients, where $0 \leq i \leq \ell+1$,

$$\begin{aligned}
\frac{t}{q}\mathtt{ct}'(s) &= \frac{t}{q}\left(c'_0 + c'_1 s + \ldots + c'_{M-1}s^{M-1}\right) \\
&= \frac{t}{q}\left[c_0 + \sum_{i=0}^{\ell} \mathtt{evk}_M[i][0]c_M^{(i)} + \left(c_1 + \sum_{i=0}^{\ell} \mathtt{evk}_M[i][1]c_M^{(i)}\right)s + \ldots + c_{M-1}s^{M-1}\right] \\
&= \frac{t}{q}\left(\sum_{i=0}^{\ell} \mathtt{evk}_M[i][0]c_M^{(i)} + s\sum_{i=0}^{\ell} \mathtt{evk}_M[i][1]c_M^{(i)}\right) + \frac{t}{q}\left(c_0 + c_1 s + \ldots + c_{M-1}s^{M-1}\right) \\
&= \frac{t}{q}\left(-\sum_{i=0}^{\ell} e_{M,i}c_M^{(i)} + \sum_{i=0}^{\ell} a_i q c_M^{(i)} + s^M \sum_{i=0}^{\ell} w^i c_M^{(i)}\right) + \frac{t}{q}\left(c_0 + c_1 s + \ldots + c_{M-1}s^{M-1}\right) \\
&= \frac{t}{q}\left(-\sum_{i=0}^{\ell} e_{M,i}c_M^{(i)} + \sum_{i=0}^{\ell} a_i q c_M^{(i)}\right) + \frac{t}{q}s^M c_M + \frac{t}{q}\left(c_0 + c_1 s + \ldots + c_{M-1}s^{M-1}\right) \\
&= -\frac{t}{q}\sum_{i=0}^{\ell} e_{M,i}c_M^{(i)} + \frac{t}{q}\left(c_0 + c_1 s + \ldots + c_{M-1}s^{M-1} + c_M s^M\right) + t\sum_{i=0}^{\ell} a_i c_M^{(i)} \\
&= m + v - \frac{t}{q}\sum_{i=0}^{\ell} e_{M,i}c_M^{(i)} + \left(a_{\ell+1} + \sum_{i=0}^{\ell} a_i c_M^{(i)}\right)t\,.
\end{aligned}$$

Hence, the noise grows by an additive factor $-\frac{t}{q}\sum_{i=0}^{\ell} e_{M,i}c_M^{(i)}$ in a one-step relinearization. Iterating this process, we find the noise after relinearization:

$$v_{relin} = v - \frac{t}{q}\sum_{j=0}^{M-N-1}\sum_{i=0}^{\ell} e_{M-j,i}c_{M-j}^{(i)}\,.$$

We can then bound $\|v_{relin}\|$ as follows:

$$\begin{aligned}\|v_{relin}\| &= \left\| v - \frac{t}{q}\sum_{j=0}^{M-N-1}\sum_{i=0}^{\ell} e_{M-j,i}c_{M-j}^{(i)} \right\| \\ &\leq \|v\| + \frac{t}{q}\sum_{j=0}^{M-N-1}\sum_{i=0}^{\ell} \left\| e_{M-j,i}c_{M-j}^{(i)} \right\| \\ &\leq \|v\| + \frac{t}{q}(\ell+1)(M-N)\delta Bw\,.\end{aligned}$$

□

Next we consider the invariant noise growth in a plain multiplication operation.

Lemma 50. *Let* $\mathtt{ct}$ *be a ciphertext encrypting* m_1 *with invariant noise* v*, and let* m_2 *be a plaintext polynomial. Let* $\mathtt{ct}_{multp}$ *denote the ciphertext obtained by plain multiplication of* $\mathtt{ct}$ *with* m_2*. Then the invariant noise in* $\mathtt{ct}_{multp}$ *is* $v_{multp} = m_2 v$*, and we have the bound*

$$\|v_{pmult}\| \leq \frac{\delta t}{2}\|v\|\,.$$

Proof: By definition the ciphertext $\mathtt{ct}_{multp} = (m_2c_0, \ldots, m_2c_k)$. For some polynomial b with integer coefficients, $[m_1m_2]_t = m_1m_2 + bt$. Hence for some polynomial a with integer coefficients:

$$\begin{aligned}\frac{t}{q}\mathtt{ct}_{multp}(s) &= \frac{t}{q}\left(m_2c_0 + m_2c_1s + \cdots + m_2c_ks^k\right) \\ &= m_2\frac{t}{q}\mathtt{ct}(s) \\ &= m_2(m_1 + v + at) \\ &= [m_1m_2]_t + m_2v + (m_2a - b)t\,.\end{aligned}$$

Hence the invariant noise is $v_{multp} = m_2v$ and we have:

$$\|v_{multp}\| = \|m_2v\| \leq \delta \cdot \|m_2\| \cdot \|v\|\,,$$

from which the claimed bound follows. □

Next we consider the invariant noise growth in a plain addition operation.

Lemma 51. *Let* $\mathtt{ct}$ *be a ciphertext encrypting* m_1 *with invariant noise* v*, and let* m_2 *be a plaintext polynomial. Let* $\mathtt{ct}_{addp}$ *denote the ciphertext obtained by plain addition of* $\mathtt{ct}$ *with* m_2*. Then the invariant noise* v_{addp} *in* $\mathtt{ct}_{addp}$ *can be bounded as*

$$\|v_{padd}\| \leq \|v\| + \frac{t^2}{q}\,.$$

Proof: By definition of plain addition we have $\mathtt{ct}_{addp} = (c_0 + \Delta m_2, c_1, \ldots, c_k)$. For some polynomial b with integer coefficients, $[m_1 + m_2]_t = m_1 + m_2 + bt$. Hence for some polynomial a with integer coefficients:

$$\begin{aligned}
\frac{t}{q}\mathtt{ct}_{addp}(s) &= \frac{t}{q}\left(c_0 + \Delta m_2 + c_1 s + \cdots + c_k s^k\right) \\
&= \frac{\Delta t}{q} m_2 + \frac{t}{q}\mathtt{ct}(s) \\
&= \frac{q - r_t(q)}{q} m_2 + m_1 + v + at \\
&= [m_1 + m_2]_t + v - \frac{r_t(q)}{q} m_2 + (a - b)t
\end{aligned}$$

Hence the noise is $v_{addp} = v - \frac{r_t(q)}{q} m_2$ and this can be bounded as:

$$\begin{aligned}
\|v_{addp}\| &\leq \|v\| + \frac{r_t(q)}{q}\|m_2\| \\
&\leq \|v\| + \frac{r_t(q) \cdot t}{q} \\
&\leq \|v\| + \frac{t^2}{q}\,.
\end{aligned}$$

□

Next we consider the invariant noise growth in a homomorphic negation operation.

Lemma 52. *Let* $\mathtt{ct}$ *be a ciphertext encrypting* m *with invariant noise* v *and* $\mathtt{ct}_{neg}$ *be its negation. The invariant noise* v_{neg} *in* $\mathtt{ct}_{neg}$ *is given by* $v_{neg} = -v$ *and we have*

$$\|v_{neg}\| = \|v\|\,.$$

Proof: If $\mathtt{ct} = (c_0, c_1, \ldots, c_k)$ then its negation $\mathtt{ct}_{neg} = (-c_0, -c_1, \ldots, -c_k)$. Note that since $\|m\| < t/2$ we have $[-m]_t = -m + bt$ for $\|b\| \leq 1$. So for some integer

coefficient polynomial a:

$$\begin{aligned}\frac{t}{q}\mathtt{ct}_{neg}(s) &= \frac{t}{q}(-(c_0, c_1, \ldots, c_k)) \\ &= -\frac{t}{q}\mathtt{ct}(s) \\ &= -m - v - at \\ &= [-m]_t - v - (b+a)t\end{aligned}$$

Hence the noise v_{neg} in $\mathtt{ct}_{neg}$ is $-v$ and $\|v_{neg}\| = \|v\|$. □

Finally, we consider the invariant noise growth in a homomorphic subtraction.

Lemma 53. *Let $\mathtt{ct}_1 = (c_0, c_1, \ldots, c_j)$ and $\mathtt{ct}_2 = (d_0, d_1, \ldots, d_k)$ be two ciphertexts encrypting m_1, m_2 and having invariant noises v_1, v_2 respectively. The invariant noise v_{sub} in their difference $\mathtt{ct}_{sub}$ is given by $v_{sub} = v_1 - v_2$ and is bounded as $\|v_{sub}\| \leq \|v_1\| + \|v_2\|$.*

Proof: By definition of homomorphic subtraction, $\mathtt{ct}_{sub}$ encrypts $[m_1 - m_2]_t$. Let $[m_1 - m_2]_t = m_1 - m_2 + a_0 t$ for some integer coefficient polynomial a_0. Suppose without loss of generality that $\max(j, k) = j$, so that

$$\mathtt{ct}_{sub} = (c_0 - d_0, \ldots, c_k - d_k, c_{k+1}, \ldots c_j)\,.$$

By definition of invariant noise in $\mathtt{ct}_1$ and $\mathtt{ct}_2$, we have

$$\begin{aligned}\frac{t}{q}\mathtt{ct}_1(s) &= m_1 + v_1 + a_1 t\,, \\ \frac{t}{q}\mathtt{ct}_2(s) &= m_2 + v_2 + a_2 t\,,\end{aligned}$$

for some polynomials a_1, a_2 with integer coefficients. Thus:

$$\begin{aligned}\frac{t}{q}\mathtt{ct}_{sub}(s) &= \frac{t}{q}\left((c_0 - d_0) + (c_1 - d_1)s + \cdots + c_j s^j\right) \\ &= \frac{t}{q}\mathtt{ct}_1(s) - \frac{t}{q}\mathtt{ct}_2(s) \\ &= [m_1 - m_2]_t + v_1 - v_2 + (a_1 - a_2 + a_0)t\,,\end{aligned}$$

so the invariant noise is $v_{sub} = v_1 - v_2$, and is bounded as

$$\|v_{sub}\| = \|v_1 - v_2\| \leq \|v_1\| + \|v_2\|\,.$$

□

7.4.3 Discussion

In Table 7.3 we summarise the inherent and invariant noise bounds presented in the previous subsections. We have seen that many of the bounds involve the ring expansion factor δ. In general δ can be bounded as $\delta \leq n$ and so the strictest noise bounds should use $\delta = n$. However, in certain situations we can estimate more accurately the value of δ and obtain tighter upper bounds on the noise growth behaviour. For example, we typically do not know the underlying plaintexts of the inputs of homomorphic evaluation operations. However, when performing a plain multiplication, we know the plaintext m_2 that forms part of the product, and in particular we know the number N of nonzero coefficients it has and its norm $\|m_2\|$. Although at worst $N = n$, in some cases N is much less than n, for example if the plaintext is the binary encoding of a small integer. The inherent noise $v_{multp_{inh}}$ in the output of a plain multiplication of a ciphertext with inherent noise v_{inh} and a plaintext m_2 contains a term $m_2 \cdot v_{inh}$, and similarly the invariant noise v_{multp} in the output of a plain multiplication of a ciphertext with invariant noise v and a plaintext m_2 is $v_{multp} = m_2 \cdot v$. In both cases when considering the noise growth we need to bound a term $\|m_2 \cdot v\|$. Since m_2 has N nonzero coefficients, the maximal coefficient of $m_2 \cdot v$ is at worst formed of N cross terms each of size equal to the maximal coefficient of m_2 multiplied by the maximal coefficient of v. So we can estimate that the expansion factor δ when multiplying by the particular element m_2 is at most N. This means we can bound $\|m_2 \cdot v\| \leq N \cdot \|m_2\| \cdot \|v\|$ and so we estimate $\delta = N$.

From Table 7.3 we conclude that the invariant noise is both more natural, and more convenient to use, than the inherent noise. For example, we can see that the invariant noise growth bounds have fewer terms and hence are simpler than their inherent noise analogues. This can be explained because the definition of inherent noise does not capture all parts of the ciphertext that could cause rounding error. As we saw in the proof of Lemma 35, what we need to bound to ensure correctness is $\left\| -\frac{r_t(q)}{t} m + v_{inh} \right\|$. This is the *critical quantity* used by Costache and Smart [76]. The inherent noise does not take into account the $-\frac{r_t(q)}{t} m$ term in the critical quantity. In turn, this term introduces additional cross terms in multiplication and plain multiplication which explain why these inherent noise bounds especially are more complicated than their invariant noise analogues. From Table 7.3 it is clear which terms in the inherent noise bounds correspond with a term in the analogous invariant noise bound and

	Inherent noise bound	Invariant noise bound
Initial	$B(1+2\delta)$	$\frac{t}{q}\|m\| + \frac{t}{q}B(1+2\delta)$
Addition	$\|v_{1_{inh}}\| + \|v_{2_{inh}}\| + t$	$\|v_1\| + \|v_2\|$
Multiplication	$+\frac{\delta^{J+1}-1}{2(\delta-1)}$ $+\left(\delta t + \frac{\delta t(\delta^{j_2+1}-1)}{2(\delta-1)} + \frac{\delta t^2}{2q}\right)\|v_{1_{inh}}\|$ $+\left(\delta t + \frac{\delta t(\delta^{j_1+1}-1)}{2(\delta-1)} + \frac{\delta t^2}{2q}\right)\|v_{2_{inh}}\|$ $+\left(\frac{3\delta t}{q}\right)\|v_{1_{inh}}\| \cdot \|v_{2_{inh}}\|$ $\frac{2\delta t^2+t}{2} - \frac{3\delta t^3}{4q} + \frac{t^2\delta(\delta^{j_2+1}+\delta^{j_1+1}-2)}{4(\delta-1)}$	$\frac{t(\delta^{J+1}-1)}{2q(\delta-1)}$ $+\left(\delta t + \frac{\delta t(\delta^{j_2+1}-1)}{2(\delta-1)}\right)\|v_1\|$ $+\left(\delta t + \frac{\delta t(\delta^{j_1+1}-1)}{2(\delta-1)}\right)\|v_2\|$ $+3\delta \cdot \|v_1\| \cdot \|v_2\|$
Relinearization	$\|v_{inh}\| + (M-N)(\ell+1)\delta Bw$	$\|v\| + \frac{t}{q}(M-N)(\ell+1)\delta Bw$
Multiply plain	$\frac{\delta t}{2}\|v_{inh}\| + \frac{t}{2} + \frac{\delta t^2}{4}$	$\frac{\delta t}{2}\|v\|$
Add plain	$\|v_{inh}\| + t$	$\|v\| + \frac{t^2}{q}$
Negation	$\|v_{inh}\| + t$	$\|v\|$
Subtraction	$\|v_{1_{inh}}\| + \|v_{2_{inh}}\| + t$	$\|v_1\| + \|v_2\|$

Table 7.3: Summary of bounds for the growth of invariant and inherent noise in ciphertexts after various homomorphic operations in SEAL.

which are extra terms arising from this $-\frac{r_t(q)}{t}m$ term. In all cases, except for the initial noise in a fresh ciphertext, these extra terms are found in the inherent noise bounds rather than the invariant noise bounds.

By definition the invariant noise captures all parts of the ciphertext that could cause rounding error. This can also be seen from the fact that the critical quantity is a scaling of the invariant noise (see Lemma 34). We believe this makes the invariant noise a very intuitive definition of noise. This intuition carries into the noise bounds. For example, the invariant noise after addition is bounded by the sum of the input noises, and the invariant noise after negation is the negation of the input noise.

7.4.4 Heuristic noise growth estimates

The strict upper bounds for the noise growth presented in Sections 7.4.1 and 7.4.2 result in poor practical estimates. Indeed, to implement noise growth estimates to enable automatic parameter selection functionality in SEAL 2.0 and SEAL v2.1 average-case heuristic inherent noise growth estimates were used. In this section we give a heuristic noise growth analysis, similar to that implemented in SEAL v2.2, using the invariant noise. These heuristic noise bounds are analogous to the estimates presented by Costache and Smart [76] using the critical quantity and are derived in the same way.

The heuristic upper bounds are obtained by using the *canonical embedding norm* $\|\cdot\|^{\text{can}}$, as used in [76, 110, 108], instead of the usual infinity norm $\|\cdot\|$ as used in Sections 7.4.1 and 7.4.2. The canonical embedding norm of an element $a \in K$ is defined to be the infinity norm of the canonical embedding of a, so $\|a\|^{\text{can}} = \|\sigma(a)\|$. We will use the following properties of the canonical embedding norm. For any polynomial a we have $\|a\| \leq \|a\|^{\text{can}} \leq \|a\|_1$ and for any polynomials a, b we have $\|ab\|^{\text{can}} \leq \|a\|^{\text{can}} \|b\|^{\text{can}}$. Since the usual infinity norm is always bounded from above by the canonical norm, and we require $\|v\| < \frac{1}{2}$ for correctness, it suffices to ensure that $\|v\|^{\text{can}} \leq \frac{1}{2}$.

A polynomial e drawn from the SEAL error distribution, which has standard deviation σ, is such the canonical embedding of e has standard deviation $\sigma_e = \sigma\sqrt{n}$. A polynomial s drawn from the SEAL secret key distribution is such that the canonical embedding of s has standard deviation $\sigma_s = \sqrt{\frac{2}{3}n}$. A polynomial $c^{(i)}$ distributed uniformly in $[-\frac{w}{2}, \frac{w}{2}]$ is such that the canonical embedding of $c^{(i)}$ has standard deviation $\sigma_{c^{(i)}} = \frac{w\sqrt{n}}{\sqrt{12}}$ and similarly a polynomial c distributed uniformly in $[-\frac{q}{2}, \frac{q}{2}]$ is such that the canonical embedding of c has standard deviation $\sigma_c = \frac{q\sqrt{n}}{\sqrt{12}}$. A polynomial ε distributed uniformly in $[-\frac{1}{2}, \frac{1}{2}]$ is such that the canonical embedding of ε has standard deviation $\sigma_\varepsilon = \frac{\sqrt{n}}{\sqrt{12}}$.

Following Costache and Smart [76] we use the following estimates: $\|a\|^{\text{can}} \leq 6\,\sigma_a$ and $\|ab\|^{\text{can}} \leq 16\,\sigma_a\,\sigma_b$ and $\|abc\|^{\text{can}} \leq 40\,\sigma_a\,\sigma_b\,\sigma_c$. This is sufficient to present the bounds below, but multiplication is not presented for ciphertexts of size larger than 2 as this would require bounding the canonical norm of a product of four or more polynomials.

We could easily produce a similar bound to those presented in [76] for the canonical norm of the product of four elements in terms of the standard deviations of the four elements (and so on), and generalise the multiplication heuristic accordingly.

We first give a heuristic bound for the initial invariant noise in a fresh ciphertext.

Lemma 54. *Let* $\mathtt{ct}$ *be a fresh encryption of a message* $m \in R_t$*. Let* N_m *be an upper bound on the number of nonzero terms in the polynomial* m*. With high probability, the invariant noise* v *in* $\mathtt{ct}$ *satisfies*

$$\|v\|^{can} \leq \frac{r_t(q)}{q} \cdot N_m \cdot \|m\| + \frac{t}{q} \cdot 2\sigma \left(\frac{16\sqrt{2}}{\sqrt{3}} n + 3\sqrt{n} \right) .$$

Proof: By Lemma 46, the invariant noise in a fresh ciphertext is

$$v = \frac{t}{q} \left(\frac{-r_t(q)}{t} m - eu + e_1 + e_2 s \right) .$$

So we can bound:

$$\begin{aligned}
\|v\|^{\text{can}} &\leq \frac{r_t(q)}{q} \|m\|^{\text{can}} + \frac{t}{q} \|eu + e_1 + e_2 s\|^{\text{can}} \\
&\leq \frac{r_t(q)}{q} \|m\|_1 + \frac{t}{q} \left(\|eu\|^{\text{can}} + \|e_1\|^{\text{can}} + \|e_2 s\|^{\text{can}} \right) \\
&\leq \frac{r_t(q)}{q} \cdot N_m \cdot \|m\| + \frac{t}{q} \left(16 \cdot \sigma\sqrt{n} \cdot \frac{\sqrt{2}}{\sqrt{3}} \sqrt{n} + 6\sigma\sqrt{n} + 16 \cdot \sigma\sqrt{n} \cdot \frac{\sqrt{2}}{\sqrt{3}} \sqrt{n} \right) \\
&\leq \frac{r_t(q)}{q} \cdot N_m \cdot \|m\| + \frac{t}{q} \cdot 2\sigma \left(\frac{16\sqrt{2}}{\sqrt{3}} n + 3\sqrt{n} \right) .
\end{aligned}$$

□

Next we give a bound for the invariant noise in homomorphic addition.

Lemma 55. *Let* $\mathtt{ct}_1$ *and* $\mathtt{ct}_2$ *be two ciphertexts encrypting* $m_1, m_2 \in R_t$*, and having invariant noises* v_1*,* v_2*, respectively. Then the invariant noise* v_{add} *in their sum* $\mathtt{ct}_{add}$ *satisfies* $\|v_{add}\|^{can} \leq \|v_1\|^{can} + \|v_2\|^{can}$*.*

Proof: By Lemma 47 the invariant noise $v_{add} = v_1 + v_2$. Hence

$$\|v_{add}\|^{\text{can}} = \|v_1 + v_2\|^{\text{can}} \leq \|v_1\|^{\text{can}} + \|v_2\|^{\text{can}} .$$

□

Next we give a heuristic bound for the invariant noise in homomorphic multiplication.

Lemma 56. *Let $\mathtt{ct}_1$ be a ciphertext of size 2 encrypting m_1 with invariant noise v_1, and let $\mathtt{ct}_2$ be a ciphertext of size 2 encrypting m_2 with invariant noise v_2. Let N_{m_1} and N_{m_2} be upper bounds on the number of nonzero terms in the polynomials m_1 and m_2, respectively. Then with high probability, the invariant noise v_{mult} in the product $\mathtt{ct}_{mult}$ satisfies the following bound:*

$$\begin{aligned}\|v_{mult}\|^{can} \leq 3\,\|v_1\|^{can}\,\|v_2\|^{can} + \frac{2t\sqrt{n}}{q\sqrt{12}}\left(3 + \frac{8\sqrt{2}}{\sqrt{3}}\sqrt{n} + \frac{40}{3}n\right) \\ \left(2N_{m_2}\|m_2\| + \frac{2t\sqrt{n}}{\sqrt{12}}\left(3 + \frac{8\sqrt{2}}{\sqrt{3}}\sqrt{n} + \frac{40}{3}n\right)\right)\|v_1\|^{can} \\ \left(2N_{m_1}\|m_1\| + \frac{2t\sqrt{n}}{\sqrt{12}}\left(3 + \frac{8\sqrt{2}}{\sqrt{3}}\sqrt{n} + \frac{40}{3}n\right)\right)\|v_2\|^{can}\end{aligned}$$

Proof: By definition of invariant noise in the input ciphertexts we have:

$$\begin{aligned}\|a_i t\|^{\text{can}} &= \left\|\frac{t}{q}\mathtt{ct}_i(s) - m_i - v_i\right\|^{\text{can}} \\ &\leq \frac{t}{q}\,\|\mathtt{ct}_i(s)\|^{\text{can}} + \|m_i\|^{\text{can}} + \|v_i\|^{\text{can}}\ .\end{aligned}$$

We can bound $\|\mathtt{ct}_i(s)\|^{\text{can}}$ as follows:

$$\begin{aligned}\|\mathtt{ct}_i(s)\|^{\text{can}} &= \left\|c_{i,0} + c_{i,1}s + c_{i,2}s^2\right\|^{\text{can}} \\ &\leq \|c_{i,0}\|^{\text{can}} + \|c_{i,1}s\|^{\text{can}} + \left\|c_{i,2}s^2\right\|^{\text{can}} \\ &\leq 6\cdot\frac{q\sqrt{n}}{\sqrt{12}} + 16\cdot\frac{q\sqrt{n}}{\sqrt{12}}\cdot\sqrt{\frac{2}{3}n} + 40\cdot\frac{q\sqrt{n}}{\sqrt{12}}\cdot\sqrt{\frac{2}{3}n}\cdot\sqrt{\frac{2}{3}n} \\ &= \frac{2q\sqrt{n}}{\sqrt{12}}\left(3 + \frac{8\sqrt{2}}{\sqrt{3}}\sqrt{n} + \frac{40}{3}n\right)\ .\end{aligned}$$

We can bound $\left\|\sum_{i=0}^{2}\epsilon_i s^i\right\|^{\text{can}}$ as follows:

$$\begin{aligned}\left\|\sum_{i=0}^{2}\epsilon_i s^i\right\|^{\text{can}} &\leq \|\epsilon_0\|^{\text{can}} + \|\epsilon_1\cdot s\|^{\text{can}} + \|\epsilon_2\cdot s\cdot s\|^{\text{can}} \\ &\leq 6\cdot\frac{\sqrt{n}}{\sqrt{12}} + 16\cdot\frac{\sqrt{n}}{\sqrt{12}}\cdot\sqrt{\frac{2}{3}n} + 40\cdot\frac{\sqrt{n}}{\sqrt{12}}\cdot\sqrt{\frac{2}{3}n}\cdot\sqrt{\frac{2}{3}n} \\ &= \frac{2\sqrt{n}}{\sqrt{12}}\left(3 + \frac{8\sqrt{2}}{\sqrt{3}}\sqrt{n} + \frac{40}{3}n\right)\ .\end{aligned}$$

By Lemma 48 the invariant noise is given by:

$$\begin{aligned}
\|v_{\text{mult}}\|^{\text{can}} &= \left\| m_1 v_2 + m_2 v_1 + v_1 v_2 + v_1 a_2 t + v_2 a_1 t + \frac{t}{q} \sum_{i=0}^{2} \epsilon_i s^i \right\|^{\text{can}} \\
&\leq \|m_1 v_2\|^{\text{can}} + \|m_2 v_1\|^{\text{can}} + \|v_1 v_2\|^{\text{can}} + \|v_1 a_2 t\|^{\text{can}} + \|v_2 a_1 t\|^{\text{can}} \\
&\quad + \frac{t}{q} \left\| \sum_{i=0}^{2} \epsilon_i s^i \right\|^{\text{can}} \\
&\leq \|m_1\|_1 \|v_2\|^{\text{can}} + \|m_2\|_1 \|v_1\|^{\text{can}} + \|v_1\|^{\text{can}} \|v_2\|^{\text{can}} \\
&\quad + \|v_1\|^{\text{can}} \left(\frac{t}{q} \left\| c_0 + c_1 s + \cdots + c_{j_2} s^{j_2} \right\|^{\text{can}} + \|m_2\|^{\text{can}} + \|v_2\|^{\text{can}}\right) \\
&\quad + \|v_2\|^{\text{can}} \left(\frac{t}{q} \left\| c_0 + c_1 s + \cdots + c_{j_1} s^{j_1} \right\|^{\text{can}} + \|m_1\|^{\text{can}} + \|v_1\|^{\text{can}}\right) \\
&\quad + \frac{t}{q} \left\| \sum_{i=0}^{2} \epsilon_i s^i \right\|^{\text{can}} \\
&\leq 2N_{m_1} \|m_1\| \, \|v_2\|^{\text{can}} + 2N_{m_2} \|m_2\| \, \|v_1\|^{\text{can}} + 3 \, \|v_1\|^{\text{can}} \, \|v_2\|^{\text{can}} \\
&\quad + \frac{2t\sqrt{n}}{\sqrt{12}} \left(3 + \frac{8\sqrt{2}}{\sqrt{3}} \sqrt{n} + \frac{40}{3} n \right) \|v_1\|^{\text{can}} \\
&\quad + \frac{2t\sqrt{n}}{\sqrt{12}} \left(3 + \frac{8\sqrt{2}}{\sqrt{3}} \sqrt{n} + \frac{40}{3} n \right) \|v_2\|^{\text{can}} \\
&\quad + \frac{2t\sqrt{n}}{q\sqrt{12}} \left(3 + \frac{8\sqrt{2}}{\sqrt{3}} \sqrt{n} + \frac{40}{3} n \right) .
\end{aligned}$$

□

Next we give a heuristic bound for the invariant noise in a relinearization operation.

Lemma 57. *Let* $\mathtt{ct}$ *be a ciphertext of size* $M+1$ *encrypting* m*, and having invariant noise* v*. Let* $\mathtt{ct}_{relin}$ *of size* $N+1$ *be the ciphertext obtained by the relinearization of* $\mathtt{ct}$*, where* $2 \leq N+1 < M+1$*. Then with high probability, the invariant noise* v_{relin} *in* $\mathtt{ct}_{relin}$ *can be bounded as*

$$\|v_{relin}\|^{can} \leq \|v\|^{can} + \frac{t}{q} \, (M-N)(\ell+1) \frac{8}{\sqrt{3}} \, \sigma \, n \, w \, .$$

Proof: By Lemma 49 the invariant noise is given by

$$v_{\text{relin}} = v - \frac{t}{q} \sum_{j=0}^{M-N-1} \sum_{i=0}^{\ell} e_{(M-j),i} c_{M-j}^{(i)} \, .$$

Thus we have:

$$\begin{aligned}
\|v_{\text{relin}}\|^{\text{can}} &= \left\| v + \frac{t}{q} \sum_{j=0}^{M-N} \sum_{i=0}^{\ell} -e_{(M-j),i} c_{M-j}^{(i)} \right\|^{\text{can}} \\
&\leq \|v\|^{\text{can}} + \frac{t}{q} \sum_{j=0}^{M-N} \sum_{i=0}^{\ell} \left\| -e_{(M-j),i} c_{M-j}^{(i)} \right\|^{\text{can}} \\
&= \|v\|^{\text{can}} + \frac{t}{q}(M-N)(\ell+1) \left\| e_{(M-j),i} c_{M-j}^{(i)} \right\|^{\text{can}} \\
&\leq \|v\|^{\text{can}} + \frac{t}{q}(M-N)(\ell+1) \cdot 16 \cdot \sigma\sqrt{n} \cdot \frac{w\sqrt{n}}{\sqrt{12}} \,.
\end{aligned}$$

□

Next we give a heuristic bound for the invariant noise in plain multiplication.

Lemma 58. *Let* $\mathtt{ct}$ *be a ciphertext encrypting* m_1 *with invariant noise* v*, and let* m_2 *be a plaintext polynomial. Let* N_{m_2} *be an upper bound on the number of nonzero terms in the polynomial* m_2*. Let* $\mathtt{ct}_{multp}$ *denote the ciphertext obtained by plain multiplication of* $\mathtt{ct}$ *with* m_2*. Then the invariant noise* v_{multp} *in* $\mathtt{ct}_{multp}$ *can be bounded as*

$$\|v_{multp}\|^{can} \leq N_{m_2} \cdot \|m_2\| \cdot \|v\|^{can} \,.$$

Proof: By Lemma 50 we have $v_{multp} = m_2 \cdot v$. Therefore we can bound

$$\begin{aligned}
\|v_{multp}\|^{\text{can}} &= \|m_2 \cdot v\|^{\text{can}} \\
&\leq \|m_2\|^{\text{can}} \cdot \|v\|^{\text{can}} \\
&\leq \|m_2\|_1 \cdot \|v\|^{\text{can}} \\
&\leq N_{m_2} \cdot \|m_2\| \cdot \|v\|^{\text{can}} \,.
\end{aligned}$$

□

Finally, we give a heuristic bound for the invariant noise in plain addition.

Lemma 59. *Let* $\mathtt{ct}$ *be a ciphertext encrypting* m_1 *with invariant noise* v*, and let* m_2 *be a plaintext polynomial. Let* $\mathtt{ct}_{addp}$ *denote the ciphertext obtained by plain addition of* $\mathtt{ct}$ *with* m_2*. Then the invariant noise* v_{addp} *in* $\mathtt{ct}_{addp}$ *can be bounded as*

$$\|v_{addp}\|^{can} \leq \|v\|^{can} + \frac{r_t(q)}{q} \cdot N_{m_2} \cdot \|m_2\| \,.$$

Proof: By Lemma 51 we have the invariant noise $v_{addp} = v - \frac{r_t(q)}{q} m_2$. Therefore we can bound:

$$\begin{aligned}
\|v_{addp}\|^{\text{can}} &= \left\| v - \frac{r_t(q)}{q} m_2 \right\|^{\text{can}} \\
&\leq \|v\|^{\text{can}} + \frac{r_t(q)}{q} \|m_2\|^{\text{can}} \\
&\leq \|v\|^{\text{can}} + \frac{r_t(q)}{q} \|m_2\|_1 \\
&\leq \|v\|^{\text{can}} + \frac{r_t(q)}{q} \cdot N_{m_2} \cdot \|m_2\| .
\end{aligned}$$

□

7.5 Parameter selection in SEAL

In Table 7.1 we saw that the user in SEAL must specify certain encryption parameters. These parameters are: `poly_modulus`, a polynomial $x^n + 1$ specifying the ring R; `coeff_modulus`, an integer q specifying the modulus in the ciphertext space; `plain_modulus`, an integer t specifying the modulus in the plaintext space; `noise_standard_deviation` and `noise_max_deviation`, which specify the error distribution via its standard deviation σ and a bound B on the maximum size of the error; and `decomposition_bit_count`, the logarithm $\log w$ of the base w used in relinearization.

The choice of encryption parameters can significantly affect the performance, capabilities, and security of the encryption scheme. Some choices of parameters may be insecure, give poor performance, yield ciphertexts that will not work with any homomorphic operations, or a combination of all of these. In this section we will describe the parameters n, q and t and their impact on performance, and justify the choice of the default parameters. We defer a discussion of the decomposition bit count, and the related parameters ℓ and w, to Section 7.6. In order to assist the user in choosing parameters for a specific computation, SEAL provides an automatic parameter selection module. This is described in Section 7.5.1.

n	q	Estimated security upper bound (log of cost of best attack)
2048	$2^{60} - 2^{14} + 1$	118.8
4096	$2^{116} - 2^{18} + 1$	121.7
8192	$2^{226} - 2^{26} + 1$	124.1
16384	$2^{435} - 2^{33} + 1$	130.2
32768	$2^{889} - 2^{54} - 2^{53} - 2^{52} + 1$	127.4

Table 7.4: Default pairs (n, q) in SEAL v2.2 and an upper bound of their estimated security, expressed as a log of the estimated cost of the the best attack according to commit f13c4a7 of the LWE estimator. (See also Table 7.2.)

Default values. Some of the encryption parameters, namely σ and B, are optional in the sense that if the user does not specify a value they will automatically be set to default values $\sigma = 3.19$ and $B = 6\sigma$. The choice of σ means that $\alpha q \approx 8$.

Although the parameters q and x^n+1, which implicitly specifies n, must be manually chosen by the user, a recommended q is provided for typical values of n. These default (n, q) pairs are given in Table 7.4.

Choosing the polynomial modulus. The polynomial modulus (`poly_modulus`) must be a polynomial of the form $x^n + 1$, where n is a power of 2. This is both for security and performance reasons. The benefit of using a larger n is that it allows for a larger q to be used without decreasing the security level. This in turn increases the maximal inherent noise and initial invariant noise budget, and thus allows for larger t to be used, which can be beneficial for encoding. The drawback of using a larger n is that it will significantly decrease performance.

Choosing the coefficient modulus. Suppose the polynomial modulus is held fixed. Then the choice of the coefficient modulus q affects two things: the noise budget in a freshly encrypted ciphertext and the security. A larger q means a larger initial noise budget but lower security.

In principle we can take q to be any integer that is not so large that the resulting Ring-LWE instance parameterised by n, q, and σ is insecure. However, there are special forms of q that can provide a huge performance benefit. If q is of the form

$2^A - B$, where B is an integer of small absolute value, then modular reduction modulo q can be sped up, yielding overall better performance. If $2n \mid (q-1)$, the *Number Theoretic Transform* (NTT) [75] for polynomial multiplications can be used, resulting in huge performance benefits. In particular SEAL uses Harvey's algorithm for NTT [125], which additionally requires that $4q \leq \beta$, where $\beta = 2^{64\lceil \log(q)/64 \rceil}$ is the *word size* of q. The reason for this is due to the modification to the *butterfly* operation suggested by Harvey [125], which could cause a wrap around modulo q if q is chosen to be any closer to the word boundary. The default choices of q in Table 7.4 are prime numbers that satisfy all of these guidelines.

Choosing the plaintext modulus. In principle, the plaintext modulus t can be any integer. However, it must be chosen large enough such that decoding works correctly: the coefficients in the underlying plaintext grow in size during homomorphic operations, and if they wrap around modulo t then decoding may fail. The degree of the underlying plaintext polynomial will also grow during homomorphic operations, so we must also ensure n is large enough. Costache *et al.* [78] give detailed bounds for n and t such that decoding will succeed.

We have seen that t must be chosen sufficiently large. However, the larger t is, the faster we would expect invariant and inherent noise to grow in certain operations, especially homomorphic multiplication (see Table 7.3). Furthermore, the inherent noise growth in certain operations also depends on the remainder $r_t(q)$ of q on division by t, which could be as large as t in general. Suppose the choice of q is fixed, then we can choose t to ensure $r_t(q)$ is a small value, which means inherent noise growth behaviour is improved. For example if t is such that $t \mid (q-1)$ then $r_t(q) = 1$.

If batching is used, the requirements on t are different. Firstly, t must be chosen large enough to ensure that the values in each slot do not wrap modulo t. We should also choose t to be a prime such that $t = 1 \mod 2n$, as this provides the maximal number of plaintext slots.

7.5.1 Automatic parameter selection in SEAL

We now describe the automatic parameter selection module in SEAL, which suggests appropriate parameter choices to the user. The user describes the computation they wish to obtain parameters for, and gives information on the plaintext polynomials: namely, the number of nonzero coefficients and a bound on the size of the nonzero coefficients. The tool simulates both the growth of the noise using heuristic estimates similar to those described in Section 7.4.4, and the growth of the coefficients in the underlying plaintext polynomials. It then tries to find optimal parameters that will support this computation in the following way. It begins by setting σ and B as the default values. Then, it chooses t as the smallest power of 2 such that the coefficients in the underlying plaintext should not wrap modulo t. Next, it chooses n and q from the default pairs as the smallest pair such that there is enough room for the expected noise growth. For this (n, q) pair, a suitable value for the decomposition bit count is searched for (finding such a value will be discussed in Section 7.6). If no such value is found, then the next (n, q) pair is tried, and so on.

7.6 When to relinearize in SEAL

In this section, we discuss the relinearization operation in SEAL. Unlike in FV, relinearization is not performed by default and so ciphertext sizes grow after multiplication operations. Indeed, in SEAL the relinearization operation is very general. A ciphertext of any size $k \geq 2$ can be relinearized down to any size j where $2 \leq j \leq k$ and relinearization need not occur immediately after a multiplication (as in FV), but rather can be performed at any time. It is natural to ask when or if it makes sense to relinearize a large ciphertext to some smaller size, and this is what we will discuss in Section 7.6.1. In general the optimal answer is not clear and will depend on the specific overall homomorphic evaluation operation we wish to perform. Relinearization affects both overall noise growth and the choice of the parameters ℓ and w. We will discuss its impact on both in Section 7.6.2.

7.6.1 Effect of relinearization on overall noise growth

Clearly, in itself, the relinearization operation introduces noise. However, a lack of relinearization can also cause larger noise. As can be seen from Table 7.3, the noise growth in a multiplication operation depends on the sizes of the ciphertexts input to the operation: the larger the input ciphertexts, the larger the noise bound. Therefore, relinearizing one or both of the input ciphertexts before performing a multiplication could result in the output ciphertext of the multiplication having less noise than would be the case if neither input is relinearized, even though the relinearization process would add noise. There is also a performance benefit when performing relinearization in advance of a subsequent multiplication: the multiplication of two large ciphertexts involves computing many cross terms, so will be slower than the multiplication of two smaller ciphertexts.

It is important to note that relinearization increases the (inherent or invariant) noise only by an additive factor, whereas multiplication increases the noise also by a multiplicative factor. Therefore, after sufficiently many multiplications have been performed, the additive increase in the noise from a relinearization can be insignificant compared to the noise already present in the ciphertext before relinearization. In this case it would only be beneficial to relinearize due to the smaller noise increase in any subsequent multiplications.

On the other hand, relinearizing after the very first multiplication is typically not an optimal strategy due to the additive factor being significantly larger than the noise that would result from a multiplication of two fresh ciphertexts. Subsequent multiplications will then build more noise on top of the (relatively large) additive factor that came from relinearization. In SEAL v2.2 [60] an example is provided illustrating that performing relinearization too early can reduce the invariant noise budget in the final result. This code is reproduced in Appendix A. Four plaintexts are encrypted to form fresh ciphertexts c_1, c_2, c_3, c_4 that each have a noise budget of approximately 97 bits. The ciphertexts are multiplied together as $(c_1 \cdot c_2) \cdot (c_3 \cdot c_4)$ and we explore the effect of intermediate relinearization on the noise budget. We can either relinearize one or both of the ciphertexts $(c_1 \cdot c_2)$ and $(c_3 \cdot c_4)$ before they are finally multiplied to form the end result, or we can perform no relinearization. When no relinearization is performed, the resulting ciphertext $(c_1 \cdot c_2) \cdot (c_3 \cdot c_4)$ has a noise

budget of around 51 bits. However, after relinearization the intermediate ciphertext $(c_1 \cdot c_2)$ has a noise budget of around 40 bits, and when the relinearization of $(c_1 \cdot c_2)$ is multiplied with the relinearization of $(c_3 \cdot c_4)$ the result has a noise budget of only around 19 bits.

7.6.2 Choosing relinearization parameters

Both the computational cost of the relinearization operation and the computational cost of evaluation key generation depend on the size of the evaluation key, which depends on the parameters w and ℓ. Since the evaluation keys are only used in relinearization, the cost of their generation can also be thought of as a computational cost of relinearization (although this can be amortised). The inherent and invariant noise growth in relinearization also depends on w and ℓ (see Lemmas 41 and 49). We now discuss the choice of these parameters. Note that computational cost of, and noise growth in, relinearization are the only things affected by the choice of w and ℓ, and hence if a user does not intend to perform relinearization then they do not need to worry about selecting an appropriate w or ℓ.

The number of terms in the decomposition into base w of a ciphertext element, and hence the number of pairs of ring elements in each evaluation key, is given by $\ell + 1 = \lfloor \log_w(q) \rfloor + 1$. Thus, the choice of ℓ is entirely determined by the choice of w (assuming q has been chosen). Typically, ℓ is a small integer greater than or equal to 2. In a computation involving relinearization, it cannot be the case that $\ell = 1$, which would occur for example when $w = q$. This would imply the size of an element in the decomposition of a ciphertext component could be as large as the ciphertext component itself, and so its norm could be anything up to q. It is therefore likely that the noise growth from a relinearization operation with these parameters would exceed the maximal noise bound, so decryption would fail. So, we must have $\ell \geq 2$. However, as ℓ increases, w gets smaller, and the size of the evaluation keys grows bigger. When the other parameters are fixed this makes relinearization and evaluation key generation slower. Therefore we would want to keep ℓ as small as possible.

Although we would desire a small ℓ, it may be the case that ℓ must be somewhat

larger than 2 in order for decryption to succeed for a given n. This motivates the question of whether it is better to use a candidate ring of dimension n and a given ℓ or to choose the next larger ring and use a smaller ℓ such as $\ell = 2$. In the automatic parameter selection module in SEAL, this is addressed as follows. For a given choice of n, the parameter $\ell \geq 2$ is chosen to be the smallest value such that decryption of the output ciphertext is expected to succeed. However, if no $\ell \leq 10$ can be found, then the next larger n is tried, and so on. It is important to note that since ℓ depends on the homomorphic evaluation operation we wish to perform (some choices may not lead to successful decryption), this choice of upper bound $\ell \leq 10$ may not be optimal in all cases. Determining whether or not there is always an optimal choice for ℓ is an interesting direction for future work.

Bibliography

[1] FV-NFLlib. `https://github.com/CryptoExperts/FV-NFLlib`.

[2] PALISADE. `https://git.njit.edu/palisade/PALISADE/`.

[3] Milton Abramowitz and Irene A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55 of *National Bureau of Standards Applied Mathematics Series*. For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C., 1964.

[4] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete Gaussian sampling: Extended abstract. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 733–742. ACM Press, June 2015.

[5] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[6] Martin R. Albrecht. Estimator for the bit security of LWE instances, 2017. Available at: `https://bitbucket.org/malb/lwe-estimator/`.

[7] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129. Springer, Heidelberg, May 2017.

[8] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Algebraic algorithms for LWE problems. *ACM Comm. Computer Algebra*, 49(2):62, 2015.

[9] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the BKW algorithm on LWE. *Designs, Codes and Cryptography*, 74:325–354, 2015.

[10] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. Algebraic algorithms for LWE. Cryptology ePrint Archive, Report 2014/1018, 2014. `http://eprint.iacr.org/2014/1018`.

[11] Martin R. Albrecht and Amit Deo. Large modulus ring-LWE >= module-LWE. Cryptology ePrint Archive, Report 2017/612, 2017. `http://eprint.iacr.org/2017/612`.

[12] Martin R. Albrecht, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Lazy modulus switching for the BKW algorithm on LWE. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 429–445. Springer, Heidelberg, March 2014.

[13] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-SVP. In Hyang-Sook Lee and Dong-Guk Han, editors, *ICISC 13*, volume 8565 of *LNCS*, pages 293–310. Springer, Heidelberg, November 2014.

[14] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving uSVP and applications to LWE. Cryptology ePrint Archive, Report 2017/815, 2017. `http://eprint.iacr.org/2017/815`.

[15] Martin R. Albrecht, Emmanuela Orsini, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. Tightly secure ring-LWE based key encapsulation with short ciphertexts. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *ESORICS 2017, Part I*, volume 10492 of *LNCS*, pages 29–46. Springer, Heidelberg, September 2017.

[16] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Mathematical Cryptology*, 9(3):169–203, 2015.

[17] Nabil Alkeilani Alkadri, Johannes Buchmann, Rachid El Bansarkhani, and Juliane Krämer. A framework to select parameters for lattice-based cryptography. Cryptology ePrint Archive, Report 2017/615, 2017. `http://eprint.iacr.org/2017/615`.

[18] Erdem Alkim, Nina Bindel, Johannes A. Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. Revisiting TESLA in the quantum random oracle model. In Tanja Lange and Tsuyoshi Takagi, editors, *PQCrypto 2017*, volume 10346 of *Lecture Notes in Computer Science*, pages 143–162. Springer, 2017.

[19] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium*, pages 327–343. USENIX Association, 2016.

[20] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 1–20. Springer, Heidelberg, August 2013.

[21] Yoshinori Aono, Le Trieu Phong, and Lihua Wang. Hardness estimation of LWE via band pruning. Cryptology ePrint Archive, Report 2015/1026, 2015. `http://eprint.iacr.org/2015/1026`.

[22] Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 789–819. Springer, Heidelberg, May 2016.

[23] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009.

[24] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. Cryptology ePrint Archive, Report 2015/1192, 2015. `http://eprint.iacr.org/2015/1192`.

[25] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, July 2011.

[26] Louis J. M. Aslett, Pedro M. Esperança, and Chris C. Holmes. A review of homomorphic encryption and software tools for encrypted statistical machine learning, 2015. http://arxiv.org/abs/1508.06574.

[27] László Babai. On Lovász' lattice reduction and the nearest lattice point problem (shortened version). In Kurt Mehlhorn, editor, *STACS '86*, volume 82 of *Lecture Notes in Computer Science*, pages 13–20. Springer, 1985.

[28] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[29] Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 322–337. Springer, Heidelberg, July 2014.

[30] Shi Bai, Steven D. Galbraith, Liangze Li, and Daniel Sheffield. Improved exponential-time algorithms for inhomogeneous-SIS. Cryptology ePrint Archive, Report 2014/593, 2014. http://eprint.iacr.org/2014/593.

[31] Jean-Claude Bajard, Julien Eynard, Anwar Hasan, and Vincent Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. Cryptology ePrint Archive, Report 2016/510, 2016. http://eprint.iacr.org/2016/510.

[32] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*, pages 175:1–175:6, 2015.

[33] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th SODA*, pages 10–24. ACM-SIAM, January 2016.

[34] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45. Springer, Heidelberg, August 1998.

[35] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime. Cryptology ePrint Archive, Report 2016/461, 2016. http://eprint.iacr.org/2016/461.

[36] Nina Bindel, Johannes Buchmann, Florian Göpfert, and Markus Schmidt. Estimation of the hardness of the Learning with Errors problem with a restricted number of samples. Cryptology ePrint Archive, Report 2017/140, 2017. http://eprint.iacr.org/2017/140.

[37] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, July 2003.

[38] Guillaume Bonnoron and Caroline Fontaine. A note on ring-LWE security in the case of fully homomorphic encryption. Cryptology ePrint Archive, Report 2016/385, 2016. http://eprint.iacr.org/2016/385.

[39] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 1006–1018. ACM Press, October 2016.

[40] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE Computer Society Press, May 2015.

[41] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 45–64. Springer, Heidelberg, December 2013.

[42] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Heidelberg, August 2012.

[43] Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in LWE-based homomorphic encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 1–13. Springer, Heidelberg, February / March 2013.

[44] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[45] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.

[46] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

[47] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011.

[48] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[49] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-ABE from LWE: Unbounded attributes and semi-adaptive security. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 363–384. Springer, Heidelberg, August 2016.

[50] Johannes Buchmann, Florian Göpfert, Rachel Player, and Thomas Wunderer. On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack. Cryptology ePrint Archive, Report 2016/089, 2016. `http://eprint.iacr.org/2016/089`.

[51] Johannes A. Buchmann, Florian Göpfert, Tim Güneysu, Tobias Oder, and Thomas Pöppelmann. High-performance and lightweight lattice-based public-

key encryption. In Richard Chow and Gökay Saldamli, editors, *ACM IoTPTS@AsiaCCS*, pages 2–9. ACM, 2016.

[52] Johannes A. Buchmann, Florian Göpfert, Rachel Player, and Thomas Wunderer. On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 16*, volume 9646 of *LNCS*, pages 24–43. Springer, Heidelberg, April 2016.

[53] Peter Campbell, Michael Groves, and Dan Shepherd. Soliloquy: A cautionary tale. In *ETSI 2nd Quantum-Safe Crypto Workshop*, Ottawa, Canada, October 6–7, 2014.

[54] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, October 2012.

[55] Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. On error distributions in ring-based LWE. Cryptology ePrint Archive, Report 2016/240, 2016. `http://eprint.iacr.org/2016/240`.

[56] Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Provably weak instances of ring-LWE revisited. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 147–167. Springer, Heidelberg, May 2016.

[57] Sanjit Chatterjee, Neal Koblitz, Alfred Menezes, and Palash Sarkar. Another look at tightness II: practical issues in cryptography. In Raphael C.-W. Phan and Moti Yung, editors, *Mycrypt 2016*, volume 10311 of *Lecture Notes in Computer Science*, pages 21–55. Springer, 2017.

[58] Hao Chen, Kim Laine, and Rachel Player. Simple Encrypted Arithmetic Library - SEAL (v2.1). Technical report, September 2016. Available at `https://www.microsoft.com/en-us/research/publication/simple-encrypted-arithmetic-library-seal-v2-1/`.

[59] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library - SEAL v2.1. Cryptology ePrint Archive, Report 2017/224, 2017. `http://eprint.iacr.org/2017/224`.

[60] Hao Chen, Kim Laine, and Rachel Player. Simple Encrypted Arithmetic Library - SEAL (v2.2). Technical report, June 2017. Available at https://www.microsoft.com/en-us/research/publication/simple-encrypted-arithmetic-library-seal-v2-2/.

[61] Hao Chen, Kim Laine, Rachel Player, and Yuhou Xia. High-precision arithmetic in homomorphic encryption. Cryptology ePrint Archive, Report 2017/809, 2017. http://eprint.iacr.org/2017/809.

[62] Hao Chen, Kristin Lauter, and Katherine E. Stange. Attacks on search RLWE. Cryptology ePrint Archive, Report 2015/971, 2015. http://eprint.iacr.org/2015/971.

[63] Hao Chen, Kristin Lauter, and Katherine E. Stange. Vulnerable galois RLWE families and improved attacks. Cryptology ePrint Archive, Report 2016/193, 2016. http://eprint.iacr.org/2016/193.

[64] Yuanmi Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD thesis, Paris 7, 2013.

[65] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2011.

[66] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates (full version). http://www.di.ens.fr/~ychen/research/Full_BKZ.pdf, 2012.

[67] Jung Hee Cheon, Kyoohyung Han, Jinsu Kim, Changmin Lee, and Yongha Son. A practical post-quantum public-key cryptosystem based on spLWE. In Seokhie Hong and Jong Hwan Park, editors, *ICISC 16*, volume 10157 of *LNCS*, pages 51–74. Springer, Heidelberg, November / December 2017.

[68] Jung Hee Cheon, Jinhyuck Jeong, Joohee Lee, and Keewoo Lee. Privacy-preserving computations of predictive medical models with minimax approximation and non-adjacent form, 2017. https://www.chi.uni-hannover.de/fileadmin/ful/mitarbeiter/brenner/WAHC17_paper_11.pdf.

[69] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. Cryptology ePrint Archive, Report 2016/421, 2016. http://eprint.iacr.org/2016/421.

[70] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. Lizard: Cut off the tail! Practical post-quantum public-key encryption from LWE and LWR. Cryptology ePrint Archive, Report 2016/1126, 2016. http://eprint.iacr.org/2016/1126.

[71] Jung Hee Cheon and Damien Stehlé. Fully homomophic encryption over the integers revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 513–536. Springer, Heidelberg, April 2015.

[72] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952.

[73] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2016.

[74] Kai-Min Chung, Daniel Dadush, Feng-Hao Liu, and Chris Peikert. On the lattice smoothing parameter problem. In *CCC 2013*, pages 230–241. IEEE Computer Society, 2013.

[75] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[76] Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 325–340. Springer, Heidelberg, February / March 2016.

[77] Anamaria Costache, Nigel P. Smart, and Srinivas Vivek. Faster homomorphic evaluation of discrete fourier transforms. Cryptology ePrint Archive, Report 2016/1019, 2017. https://eprint.iacr.org/2016/1019.pdf.

[78] Anamaria Costache, Nigel P. Smart, Srinivas Vivek, and Adrian Waller. Fixed-point arithmetic in SHE schemes. Cryptology ePrint Archive, Report 2016/250, 2016. https://eprint.iacr.org/2016/250.

[79] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 559–585. Springer, Heidelberg, May 2016.

[80] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-SVP. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 324–348. Springer, Heidelberg, May 2017.

[81] Eric Crockett and Chris Peikert. Challenges for ring-LWE. Cryptology ePrint Archive, Report 2016/782, 2016. http://eprint.iacr.org/2016/782.

[82] Eric Crockett and Chris Peikert. $\Lambda \circ \lambda$ functional lattice cryptography. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 993–1005. ACM Press, October 2016.

[83] Eric Crockett and Chris Peikert. $\Lambda o \lambda$: Functional lattice cryptography. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM SIGSAC*, pages 993–1005. ACM, 2016.

[84] Wei Dai, Yarkın Doröz, Yuriy Polyakov, Kurt Rohloff, Hadi Sajjadpour, Erkay Savaş, and Berk Sunar. Implementation and evaluation of a lattice-based key-policy ABE scheme. Cryptology ePrint Archive, Report 2017/601, 2017. http://eprint.iacr.org/2017/601.

[85] Wei Dai and Berk Sunar. cuHE: A homomorphic encryption accelerator library. In Enes Pasalic and Lars R. Knudsen, editors, *BalkanCryptSec 2015*, volume 9540 of *Lecture Notes in Computer Science*, pages 169–186. Springer, 2015.

[86] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, August 2012.

[87] Alex Davidson. Obfuscation of bloom filter queries from ring-LWE. Cryptology ePrint Archive, Report 2017/448, 2017. http://eprint.iacr.org/2017/448.

[88] The FPLLL development team. fplll, a lattice reduction library. Available at https://github.com/fplll/fplll, 2016.

[89] Jintai Ding. New cryptographic constructions using generalized learning with errors problem. Cryptology ePrint Archive, Report 2012/387, 2012. http://eprint.iacr.org/2012/387.

[90] Jintai Ding, Xiang Xie, and Xiaodong Lin. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688, 2012. http://eprint.iacr.org/2012/688.

[91] Yarkin Doröz, Yin Hu, and Berk Sunar. Homomorphic AES evaluation using the modified LTV scheme. *Designs, Codes and Cryptography*, 80(2):333–358, 2016.

[92] Nico Döttling and Jörn Müller-Quade. Lossy codes and a new variant of the learning-with-errors problem. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2013.

[93] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Manual for using homomorphic encryption for bioinformatics. *Proceedings of the IEEE*, 105(3):552–567, 2017.

[94] Alexandre Duc, Florian Tramèr, and Serge Vaudenay. Better algorithms for LWE and LWR. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 173–202. Springer, Heidelberg, April 2015.

[95] Léo Ducas and Alain Durmus. Ring-LWE in polynomial rings. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 34–51. Springer, Heidelberg, May 2012.

[96] Léo Ducas-Binda. *Signatures Fondées sur les Réseaux Euclidiens: Attaques, Analyses et Optimisations*. PhD thesis, École Normale Supérieure Paris, 2013. http://cseweb.ucsd.edu/~lducas/Thesis/index.html.

[97] Kirsten Eisenträger, Sean Hallgren, and Kristin E. Lauter. Weak instances of PLWE. In Antoine Joux and Amr M. Youssef, editors, *SAC 2014*, volume 8781 of *LNCS*, pages 183–194. Springer, Heidelberg, August 2014.

[98] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

[99] Yara Elias, Kristin E. Lauter, Ekin Ozman, and Katherine E. Stange. Provably weak instances of ring-LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 63–92. Springer, Heidelberg, August 2015.

[100] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `http://eprint.iacr.org/2012/144`.

[101] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44(170):463–471, 1985.

[102] S. D. Galbraith. Space-efficient variants of cryptosystems based on learning with errors. `https://www.math.auckland.ac.nz/~sgal018/compact-LWE.pdf`, 2012.

[103] Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 207–216. ACM Press, May 2008.

[104] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, Heidelberg, April 2008.

[105] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278. Springer, Heidelberg, May 2010.

[106] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[107] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[108] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482. Springer, Heidelberg, April 2012.

[109] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. Cryptology ePrint Archive, Report 2012/099, 2012. `http://eprint.iacr.org/2012/099`.

[110] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, Heidelberg, August 2012.

[111] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[112] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[113] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *33rd ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 201–210. JMLR.org, 2016.

[114] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *ICS 2010*, pages 230–240. Tsinghua University Press, 2010.

[115] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[116] Florian Göpfert. *Securely Instantiating Cryptographic Schemes Based on the Learning with Errors Assumption.* PhD thesis, Technische Universität, Darmstadt, 2016.

[117] Florian Göpfert, Christine van Vredendaal, and Thomas Wunderer. A hybrid lattice basis reduction and quantum search attack on LWE. In Tanja Lange and Tsuyoshi Takagi, editors, *PQCrypto 2017*, volume 10346 of *Lecture Notes in Computer Science*, pages 184–202. Springer, 2017.

[118] G. Grimmett and D. Stirzaker. *Probability And Random Processes.* Oxford University Press, third edition, 2001.

[119] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 530–547. Springer, Heidelberg, September 2012.

[120] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 23–42. Springer, Heidelberg, August 2015.

[121] Shai Halevi and Victor Shoup. Design and Implementation of a Homomorphic-Encryption Library. `http://people.csail.mit.edu/shaih/pubs/he-library.pdf`, 2013.

[122] Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2014.

[123] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 641–670. Springer, Heidelberg, April 2015.

[124] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 447–464. Springer, Heidelberg, August 2011.

[125] David Harvey. Faster arithmetic for number-theoretic transforms. *Journal of Symbolic Computation*, 60:113–119, 2014.

[126] Charles Hermite. Extraits de lettres de M. Ch. Hermite à M. Jacobi sur différents objects de la théorie des nombres. (Continuation). *Journal für die reine und angewandte Mathematik*, 1850(40):279–315, 1850.

[127] Charles Hermite. *Œuvres de Charles Hermite*. Paris: Gauthier-Villars, 1905.

[128] Gottfried Herold, Elena Kirshanova, and Alexander May. On the asymptotic complexity of solving LWE. *Designs, Codes and Cryptography*, Jan 2017.

[129] Gottfried Herold and Alexander May. LP solutions of vectorial integer subset sums — cryptanalysis of Galbraith's binary matrix LWE. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 3–15. Springer, Heidelberg, March 2017.

[130] Philip S. Hirschhorn, Jeffrey Hoffstein, Nick Howgrave-Graham, and William Whyte. Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09*, volume 5536 of *LNCS*, pages 437–455. Springer, Heidelberg, June 2009.

[131] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *ANTS-III*, pages 267–288, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[132] Nick Howgrave-Graham. Approximate integer common divisors. In Joseph H. Silverman, editor, *CaLC 2001*, volume 2146 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2001.

[133] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 150–169. Springer, Heidelberg, August 2007.

[134] Antoine Joux. *Algorithmic Cryptanalysis*. Chapman & Hall/CRC, 1st edition, 2009.

[135] Erich Kaltofen. On the complexity of finding short vectors in integer lattices. *Computer Algebra*, pages 236–244, 1983.

[136] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, August 1987.

[137] Michael J. Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. On the learnability of discrete distributions. In *26th ACM STOC*, pages 273–282. ACM Press, May 1994.

[138] Miran Kim and Kristin Lauter. Private genome analysis through homomorphic encryption. *BMC Medical Informatics and Decision Making*, 15(5):S3, Dec 2015.

[139] Paul Kirchner and Pierre-Alain Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 43–62. Springer, Heidelberg, August 2015.

[140] Aleksandr Korkine and Yegor Zolotarev. Sur les formes quadratiques. *Mathematische Annalen*, 6(3):366–389, 1873.

[141] Thijs Laarhoven. *Search problems in cryptography: from fingerprinting to lattice sieving*. PhD thesis, Technische Universiteit Eindhoven, 2015. `http://repository.tue.nl/837539`.

[142] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 3–22. Springer, Heidelberg, August 2015.

[143] Thijs Laarhoven and Benne de Weger. Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 101–118. Springer, Heidelberg, August 2015.

[144] Jeffrey C Lagarias, Hendrik W Lenstra, and Claus-Peter Schnorr. Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10(4):333–348, 1990.

[145] Kim Laine and Kristin Lauter. Key recovery for LWE in polynomial time. Cryptology ePrint Archive, Report 2015/176, 2015. `http://eprint.iacr.org/2015/176`.

[146] Kim Laine and Rachel Player. Simple Encrypted Arithmetic Library - SEAL (v2.0). Technical report, September 2016. Available at `https://www.microsoft.com/en-us/research/publication/simple-encrypted-arithmetic-library-seal-v2-0/`.

[147] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, 2015.

[148] A.K. Lenstra, Jr. Lenstra, H.W., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[149] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT 14*, volume 8469 of *LNCS*, pages 318–335. Springer, Heidelberg, May 2014.

[150] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011.

[151] Cong Ling, Shuiyin Liu, Laura Luzzi, and Damien Stehlé. Decoding by embedding: Correct decoding radius and DMT optimality. In Alexander Kuleshov, Vladimir Blinovsky, and Anthony Ephremides, editors, *IEEE ISIT 2011*, pages 1106–1110. IEEE, 2011.

[152] Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *LNCS*, pages 293–309. Springer, Heidelberg, February / March 2013.

[153] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

[154] László Lovász. *An algorithmic theory of numbers, graphs and convexity*. CBMS-NSF regional conference series in applied mathematics. Philadelphia, Pa. Society for Industrial and Applied Mathematics, 1986.

[155] Vadim Lyubashevsky. Lattice signatures without trapdoors. Cryptology ePrint Archive, Report 2011/537, 2011. `http://eprint.iacr.org/2011/537`.

[156] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 577–594. Springer, Heidelberg, August 2009.

[157] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May 2010.

[158] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. Cryptology ePrint Archive, Report 2012/230, 2012. `http://eprint.iacr.org/2012/230`.

[159] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, Heidelberg, May 2013.

[160] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. Cryptology ePrint Archive, Report 2013/293, 2013. `http://eprint.iacr.org/2013/293`.

[161] Kurt Mahler. A theorem on inhomogeneous diophantine inequalities. *Nederl. Akad. Wetensch., Proc. 41*, pages 634–637, 1938.

[162] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Jet Propulsion Laboratory DSN Progress Report*, 42-44:114–116, 1978.

[163] Carlos Aguilar Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrède Lepoint. NFLlib: NTT-based fast lattice library. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 341–356. Springer, Heidelberg, February / March 2016.

[164] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 218–238. Springer, Heidelberg, August 1990.

[165] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In Phillip Rogaway, editor,

CRYPTO 2011, volume 6841 of *LNCS*, pages 465–484. Springer, Heidelberg, August 2011.

[166] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39. Springer, Heidelberg, August 2013.

[167] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

[168] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer, Berlin, Heidelberg, New York, 2009.

[169] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. *SIAM Journal on Computing*, 42(3):1364–1391, 2013.

[170] Daniele Micciancio and Michael Walter. Fast lattice point enumeration with minimal overhead. In Piotr Indyk, editor, *26th SODA*, pages 276–294. ACM-SIAM, January 2015.

[171] Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 820–849. Springer, Heidelberg, May 2016.

[172] Hermann Minkowski. *Geometrie der Zahlen.* Leipzig and Berlin: Teubner, 1910.

[173] Sean Murphy and Rachel Player. Noise distributions in homomorphic Ring-LWE, 2017. In submission.

[174] Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In Christian Cachin and Thomas Ristenpart, editors, *ACM CCSW 2011*, pages 113–124. ACM, 2011.

[175] Arnold Neumaier and Damien Stehlé. Faster lll-type reduction of lattice bases. In Sergei A. Abramov, Eugene V. Zima, and Xiao-Shan Gao, editors, *ACM ISSAC 2016*, pages 373–380. ACM, 2016.

[176] Phong Q. Nguyen and Damien Stehlé. Floating-point LLL revisited. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 215–233. Springer, Heidelberg, May 2005.

[177] Phong Q. Nguyen and Damien Stehlé. LLL on the average. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *Algorithmic Number Theory, 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006, Proceedings*, volume 4076 of *Lecture Notes in Computer Science*, pages 238–256. Springer, 2006.

[178] Phong Q. Nguyen and Damien Stehlé. Low-dimensional lattice basis reduction revisited. *ACM Trans. Algorithms*, 5(4):46:1–46:48, 2009.

[179] Phong Q. Nguyen and Brigitte Vallée, editors. *The LLL Algorithm - Survey and Applications*. Information Security and Cryptography. Springer, 2010.

[180] NIST. Call for submissions for quantum-resistant public-key cryptographic algorithms, 2016. `http://csrc.nist.gov/groups/ST/post-quantum-crypto/index.html`.

[181] Andrew Novocin, Damien Stehlé, and Gilles Villard. An lll-reduction algorithm with quasi-linear time complexity: extended abstract. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 403–412. ACM Press, June 2011.

[182] Frank WJ Olver. *NIST handbook of mathematical functions*. Cambridge University Press, 2010.

[183] Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 525–542. Springer, Heidelberg, August 2011.

[184] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

[185] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009.

[186] Chris Peikert. Lattice cryptography for the internet. In Michele Mosca, editor, *PQCrypto 2014*, volume 8772 of *Lecture Notes in Computer Science*, pages 197–219. Springer, 2014.

[187] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016.

[188] Chris Peikert. How (not) to instantiate ring-LWE. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 411–430. Springer, Heidelberg, August / September 2016.

[189] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 461–473. ACM Press, June 2017.

[190] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008.

[191] Krzysztof Pietrzak. Subspace LWE. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 548–563. Springer, Heidelberg, March 2012.

[192] Xavier Pujol and Damien Stehlé. Solving the shortest lattice vector problem in time $2^{2.465n}$. Cryptology ePrint Archive, Report 2009/605, 2009. `http://eprint.iacr.org/2009/605`.

[193] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Heidelberg, August 1992.

[194] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[195] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, September 2009.

[196] Oded Regev. The learning with errors problem (invited survey). In *IEEE CCC 2010*, pages 191–204. IEEE Computer Society, 2010.

[197] George W. Reitwiesner. Binary arithmetic. *Advances in Computers*, 1:231–308, 1960.

[198] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[199] Miruna Rosca, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Middle-product learning with errors. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 283–297. Springer, Heidelberg, August 2017.

[200] Claus-Peter Schnorr. A more efficient algorithm for lattice basis reduction. *Journal of algorithms*, 9(1):47–62, 1988.

[201] Claus-Peter Schnorr. Lattice reduction by random sampling and birthday methods. In *STACS*, volume 2607 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2003.

[202] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994.

[203] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

[204] V. Shoup. Number theory library 5.5.2 (ntl) for c++. `http://www.shoup.net/ntl/`.

[205] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1):57–81, 2014.

[206] Damien Stehlé. *Floating-Point LLL: Theoretical and Practical Aspects*, pages 179–213. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[207] Damien Stehlé. An introduction to lattice reduction. ECRYPT II Summer School on Lattices, Porto, 2012.

[208] Damien Stehlé. An overview of lattice reduction algorithms. Invited talk at ICISC, 2013.

[209] Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 27–47. Springer, Heidelberg, May 2011.

[210] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, Heidelberg, December 2009.

[211] Joop van de Pol and Nigel P. Smart. Estimating key sizes for high dimensional lattice-based systems. In Martijn Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 290–303. Springer, Heidelberg, December 2013.

[212] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 24–43. Springer, Heidelberg, May 2010.

[213] Michael Walter. Lattice point enumeration on block reduced bases. In Anja Lehmann and Stefan Wolf, editors, *ICITS 15*, volume 9063 of *LNCS*, pages 269–282. Springer, Heidelberg, May 2015.

[214] Thomas Wunderer. Revisiting the hybrid attack: Improved analysis and refined security estimates. Cryptology ePrint Archive, Report 2016/733, 2016. `http://eprint.iacr.org/2016/733`.

[215] Thomas Wunderer. Source code for "Revisiting the Hybrid Attack: Improved Analysis and Refined Security Estimates", 2016. `https://www.cdc.informatik.tu-darmstadt.de/cdc/personen/thomas-wunderer/`.

[216] Jiang Zhang, Zhenfeng Zhang, Jintai Ding, Michael Snook, and Özgür Dagdelen. Authenticated key exchange from ideal lattices. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 719–751. Springer, Heidelberg, April 2015.

Appendix A

SEAL code used in Section 7.6

```
void example_relinearization_part1()
{
    cout << "Example: early relinearization effect on noise." << endl;

    // Set up encryption parameters
    EncryptionParameters parms;
    parms.set_poly_modulus("1x^4096 + 1");
    parms.set_coeff_modulus(
                ChooserEvaluator::default_parameter_options().at(4096));
    parms.set_plain_modulus(1 << 8);
    parms.set_decomposition_bit_count(58);

    // Validate the parameters
    parms.validate();

    // Generate keys
    KeyGenerator generator(parms);
    generator.generate(1);
    Ciphertext public_key = generator.public_key();
    Plaintext secret_key = generator.secret_key();
    EvaluationKeys evaluation_keys = generator.evaluation_keys();

    // Encrypt plaintexts to generate the four fresh ciphertexts
    Plaintext plain1("5");
    Plaintext plain2("6");
    Plaintext plain3("7");
    Plaintext plain4("8");
    Encryptor encryptor(parms, public_key);
    Ciphertext enc1 = encryptor.encrypt(plain1);
    Ciphertext enc2 = encryptor.encrypt(plain2);
    Ciphertext enc3 = encryptor.encrypt(plain3);
    Ciphertext enc4 = encryptor.encrypt(plain4);

    // We need a Decryptor to be able to measure the invariant noise
    Decryptor decryptor(parms, secret_key);

    // What are the noise budgets?
    cout << "Noise budgets in the four fresh ciphertexts: "
         << decryptor.invariant_noise_budget(enc1) << " bits, "
         << decryptor.invariant_noise_budget(enc2) << " bits, "
         << decryptor.invariant_noise_budget(enc3) << " bits, "
         << decryptor.invariant_noise_budget(enc4) << " bits"
```

```
        << endl;

    // Construct an Evaluator
    Evaluator evaluator(parms, evaluation_keys);

    // Perform first part of computation
    Ciphertext enc_prod1 = evaluator.multiply(enc1, enc2);
    Ciphertext enc_prod2 = evaluator.multiply(enc3, enc4);

    // First compute the result with no relinearization
    cout << endl;
    cout << "Path 1: No relinearization" << endl;

    // Compute product of all four
    cout << "Computing result as enc_prod1*enc_prod2 ..." << endl;
    Ciphertext enc_res = evaluator.multiply(enc_prod1, enc_prod2);

    // Now enc_res has size 5
    cout << "Size of enc_res: " << enc_res.size() << endl;

    // How much noise budget are we left with?
    int nb_norelin = decryptor.invariant_noise_budget(enc_res);
    cout << "Noise budget in enc_result: " << nb_norelin
        << " bits" << endl;

    // Now compute the result with relinearization
    cout << endl;
    cout << "Path 2: With relinearization" << endl;

    // Intermediate relinearization of enc_prod1 and enc_prod2
    cout << "Relinearizing enc_prod1 and enc_prod2 to size 2 ..."
        << endl;
    Ciphertext enc_relin_prod1 = evaluator.relinearize(enc_prod1);
    Ciphertext enc_relin_prod2 = evaluator.relinearize(enc_prod2);

    // What is the noise budget after intermediate relinearization?
    cout << "Noise budgets in enc_relin_prod1: "
        << decryptor.invariant_noise_budget(enc_relin_prod1)
        << " bits, and in enc_relin_prod2: "
        << decryptor.invariant_noise_budget(enc_relin_prod2)
        << " bits " << endl;

    // Now multiply the relinearized products together
    cout << "Computing enc_res as enc_relin_prod1*enc_relin_prod2 ..."
        << endl;
    enc_res = evaluator.multiply(enc_relin_prod1, enc_relin_prod2);

    // Now enc_res has size 3
    cout << "Size of enc_res: " << enc_res.size() << endl;

    // How much noise budget are we left with?
    int nb_relin = decryptor.invariant_noise_budget(enc_res);
    cout << "Noise budget in enc_res: " << nb_relin
        << " bits" << endl;

}
```